

Ostbayerische Technische Hochschule Amberg-Weiden  
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Master Künstliche Intelligenz

**Masterarbeit**

von

**Sebastian Steindl**

**Untersuchung von Ensemble-Learning zur  
Krankheitsklassifikation auf Thorax-Röntgenbildern**

**Evaluation of ensemble learning for disease classification  
on chest radiographs**



Ostbayerische Technische Hochschule Amberg-Weiden  
Fakultät Elektrotechnik, Medien und Informatik

# Studiengang Master Künstliche Intelligenz

# Masterarbeit

von

Sebastian Steindl

# Untersuchung von Ensemble-Learning zur Krankheitsklassifikation auf Thorax-Röntgenbildern

# Evaluation of ensemble learning for disease classification on chest radiographs

Bearbeitungszeitraum: von 19.05.2021  
bis 18.11.2021

1. Prüfer: Prof. Dr. Brunner  
2. Prüfer: Prof. Dr. Tatyana Ivanovska



Bestätigung gemäß § 12 APO

---

Name und Vorname  
der Studentin/des Studenten: **Steindl, Sebastian**

Studiengang: **Master Künstliche Intelligenz**

---

Ich bestätige, dass ich die Masterarbeit mit dem Titel:

**Untersuchung von Ensemble-Learning zur Krankheitsklassifikation auf  
Thorax-Röntgenbildern**

selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

---

Datum: **29. Oktober 2021**

Unterschrift:

---

## Masterarbeit Zusammenfassung

---

Studentin/Student (Name, Vorname):

**Steindl, Sebastian**

Studiengang:

Master Künstliche Intelligenz

Aufgabensteller, Professor:

Prof. Dr. Brunner

Durchgeführt in (Firma/Behörde/Hochschule): Capgemini Germany, Nürnberg

Betreuer in Firma/Behörde:

Dr. Eldar Sultanow

Ausgabedatum: **19.05.2021**

Abgabedatum: **18.11.2021**

---

Titel:

**Untersuchung von Ensemble-Learning zur Krankheitsklassifikation auf  
Thorax-Röntgenbildern**

---

Zusammenfassung:

In dieser Arbeit sollte der Einsatz von Deep Learning und Ensembling-Techniken zur Klassifikation von Thorax-Röntgenbildern untersucht werden. Die Arbeit liefert zunächst eine Einführung in die Problemstellung, die Domäne und das Forschungsgebiet. Um das Ensembling zu realisieren wurden verschiedene Methoden aus der Literatur analysiert, implementiert und kombiniert. Diese Ansätze, die auf unterschiedliche Art versuchen, die Klassifizierungsergebnisse zu verbessern, wurden anhand extensiver Experimente erprobt. Über einen Algorithmus zur Ensemble-Selektion wurde aus der gebildeten Modellbibliothek ein Ensemble gebildet. Dieses wurde bei der CheXpert Competition eingereicht und erzielte eine ROC-AUC von 0.917 auf den geheimen Testdaten. Die Arbeit zeigt außerdem die Limitationen derartiger Assistenzsysteme zur Krankheitsdiagnose auf.

Schlüsselwörter: Thorax-Röntgenbild-Klassifikation, Ensemble-Learning, ChestX-Ray, medizinische KI, Computer-aided diagnosis, CheXpert

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Zielsetzung . . . . .	2
1.3 Aufbau der Arbeit . . . . .	2
<b>2 Grundlagen</b>	<b>4</b>
2.1 Thorax-Röntgenbilder . . . . .	4
2.2 Bestehende KI-Systeme für Thorax-Röntgenbilder . . . . .	5
2.3 Bildaugmentierung . . . . .	6
2.4 Neuronale Netze für die Bildklassifikation . . . . .	7
2.4.1 Funktionsweise von Convolutional Neural Networks . . . . .	7
2.4.2 ResNet . . . . .	10
2.4.3 DenseNet . . . . .	11
2.5 Ensemble Learning . . . . .	15
2.5.1 Mischung der Trainingsdaten . . . . .	16
2.5.2 Mischung der Modelle . . . . .	17
2.5.3 Mischung der Kombinationen . . . . .	18
2.5.4 Committee Machines . . . . .	19
2.5.5 Modell-Auswahl . . . . .	22
2.6 Wichtige Metriken . . . . .	23
2.6.1 Vom Modelloutput zur Klasse . . . . .	23
2.6.2 Konfusionsmatrix . . . . .	25
2.6.3 Genauigkeit . . . . .	26
2.6.4 True Positive Rate . . . . .	27
2.6.5 False Positive Rate . . . . .	27
2.6.6 Spezifität . . . . .	28
2.6.7 Receiver Operating Characteristic (ROC) . . . . .	28
2.6.8 Area Under the Curve (AUC) . . . . .	29
2.6.9 Erweiterung auf Multi-Class Probleme . . . . .	31
<b>3 Multi-Label Klassifizierung</b>	<b>33</b>
3.1 Klassifizierungsarten . . . . .	33
3.1.1 Binärklassifikation . . . . .	34
3.1.2 Multi-Class Klassifikation . . . . .	34
3.1.3 Multi-Label Klassifikation . . . . .	34
3.1.4 Hierarchische Klassifikation . . . . .	35

3.2	Problemtransformationen . . . . .	35
3.2.1	Binary Relevance Decomposition . . . . .	35
3.2.2	Label Powerset Transformation . . . . .	36
3.2.3	Classifier Chain . . . . .	36
3.2.4	RAkEL . . . . .	37
<b>4</b>	<b>Transfer Learning</b>	<b>38</b>
4.1	ImageNet Large Scale Visual Recognition Challenge (ILSVRC) . . . . .	39
4.2	Transfer Learning für die Bildverarbeitung . . . . .	39
4.3	Transfer Learning für medizinische Bilddaten . . . . .	41
<b>5</b>	<b>Datensätze</b>	<b>43</b>
5.1	ChestX-ray14 . . . . .	43
5.2	MIMIC-CXR . . . . .	44
5.3	PadChest . . . . .	44
5.4	CheXpert . . . . .	45
5.4.1	Beschreibung der Krankheitsbilder der Validierungsdaten . . . . .	48
5.4.2	Entscheidung für den Datensatz . . . . .	50
<b>6</b>	<b>Verwandte Arbeiten</b>	<b>51</b>
6.1	Deep Learning für Thorax-Röntgenbilder . . . . .	51
6.1.1	Automated abnormality classification of chest radiographs using deep convolutional neural networks . . . . .	52
6.1.2	Assessment of convolutional neural networks for automated classification of chest radiographs . . . . .	53
6.1.3	Weakly Supervised Lesion Localization With Probabilistic-CAM Pooling . . . . .	54
6.1.4	When does bone suppression and lung field segmantation improve Chest X-Ray disease classification . . . . .	55
6.1.5	Zusammenfassung dieser Resultate . . . . .	56
6.2	Entwicklungen während der COVID-19 Pandemie . . . . .	56
<b>7</b>	<b>Ausgewählte Methoden</b>	<b>58</b>
7.1	FRODO: Free rejection of out-of-distribution samples . . . . .	58
7.1.1	Nutzung von Ensembling-Methoden . . . . .	60
7.2	Automated Triaging of Adult Chest Radiographs . . . . .	60
7.2.1	Datensatz, Label, Bildaugmentierung . . . . .	60
7.2.2	Prioritätsprognose . . . . .	61
7.2.3	Nutzung von Ensembling-Methoden . . . . .	62
7.2.4	Resultate . . . . .	63
7.2.5	Limitationen . . . . .	63
7.3	CNNs that exploit hierarchical disease dependencies . . . . .	64
7.3.1	Label-Smoothing-Regularisierung (LSR) . . . . .	64
7.3.2	Conditional Training . . . . .	64
7.3.3	Datenvorbereitung . . . . .	66
7.3.4	Modelltraining . . . . .	66

7.3.5	Nutzung von Ensembling-Methoden . . . . .	67
7.3.6	Resultate . . . . .	67
7.3.7	Limitationen . . . . .	68
7.4	Diagnose like a radiologist . . . . .	68
7.4.1	Architektur . . . . .	68
7.4.2	Erzeugen lokaler Bildausschnitte . . . . .	69
7.4.3	Durchgeführte Experimente . . . . .	70
7.4.4	Resultate . . . . .	72
7.4.5	Nutzung von Ensembling-Methoden . . . . .	72
7.5	A Novel Approach for Multi-Label Chest X-Ray Classification . . . . .	73
7.5.1	Resultate . . . . .	74
7.5.2	Nutzung von Ensembling-Methoden . . . . .	74
7.6	Boosted cascaded convnets. . . . .	74
7.6.1	Binary Relevance und PWE Loss . . . . .	74
7.6.2	Boosted cascaded convnets . . . . .	75
7.6.3	Implementierung . . . . .	76
7.6.4	Resultate . . . . .	76
7.6.5	Nutzung von Ensembling-Methoden . . . . .	77
7.7	Robust Deep AUC Maximization . . . . .	77
7.7.1	Nutzung von Ensembling-Methoden . . . . .	78
7.8	Modell-Selektion nach AUC . . . . .	79
7.9	Intra-Epoch Evaluation . . . . .	79
<b>8</b>	<b>Implementierung</b> . . . . .	<b>81</b>
8.1	FRODO: Free rejection of out-of-distribution samples . . . . .	81
8.2	Automated Triaging of Adult Chest Radiographs . . . . .	85
8.2.1	Template Matching . . . . .	86
8.2.2	Unterschiedliche Bildauflösungen . . . . .	86
8.3	CNNs that exploit hierarchical disease dependencies . . . . .	86
8.3.1	Label-Smoothing-Regularisierung (LSR) . . . . .	86
8.3.2	Conditional Training . . . . .	88
8.4	Diagnose like a radiologist . . . . .	88
8.5	A Novel Approach for Multi-Label Chest X-Ray Classification . . . . .	89
8.6	Boosted cascaded convnets . . . . .	91
8.7	Robust Deep AUC Maximization . . . . .	92
8.8	Weitere Implementierungen . . . . .	92
8.8.1	CLAHE . . . . .	92
8.8.2	Behandlung der Daten-Imbalance . . . . .	93
<b>9</b>	<b>Durchgeführte Experimente</b> . . . . .	<b>97</b>
9.1	Vorbereitungen . . . . .	97
9.1.1	Datensatz ausbalancieren . . . . .	97
9.1.2	Datensatz an Validierungsdaten angleichen . . . . .	98
9.1.3	Template Matching . . . . .	100
9.1.4	Trainingsloop . . . . .	101
9.1.5	Patientensensitive Datenaufteilung . . . . .	101

9.2	Grundlage . . . . .	101
9.3	Erweiterungen . . . . .	102
9.4	Gewichtsinitialisierung . . . . .	103
9.5	Diagnose like a radiologist . . . . .	106
9.6	CNNs that exploit hierarchical disease dependencies . . . . .	107
9.7	CNN als Feature-Extractor . . . . .	110
9.8	Snapshot Ensemble . . . . .	112
9.9	Boosted cascaded convnets . . . . .	113
9.10	Robust Deep AUC Maximization . . . . .	114
<b>10</b>	<b>Ensembling der Methoden</b>	<b>117</b>
10.1	Ensemble-Selektion . . . . .	117
10.2	Modellbibliothek . . . . .	117
10.3	Analyse der Korrelation . . . . .	120
10.4	Finales Ensemble . . . . .	120
<b>11</b>	<b>Einreichung bei der CheXpert-Competition</b>	<b>125</b>
<b>12</b>	<b>Evaluierung auf externen Daten</b>	<b>127</b>
<b>13</b>	<b>Resultate</b>	<b>129</b>
<b>14</b>	<b>Limitationen</b>	<b>132</b>
14.1	Technische Limitationen . . . . .	132
14.2	Praxisbezogene Limitationen . . . . .	133
14.2.1	Datensatz . . . . .	133
14.2.2	Einsatz in Krankenhäusern . . . . .	134
14.2.3	Rechtliche Limitationen . . . . .	135
<b>15</b>	<b>Zusammenfassung und Ausblick</b>	<b>136</b>
<b>Literaturverzeichnis</b>		<b>138</b>
<b>Abbildungsverzeichnis</b>		<b>148</b>
<b>Tabellenverzeichnis</b>		<b>150</b>
<b>Listingverzeichnis</b>		<b>151</b>
<b>A</b>	<b>Weitere Informationen</b>	<b>152</b>
A.1	Pytorch . . . . .	152
A.1.1	Tensoren . . . . .	152
A.1.2	Autograd . . . . .	153
A.1.3	Optimisierer . . . . .	153
A.1.4	nn.Module . . . . .	153
A.1.5	Aktivierungs- und Lossfunktionen . . . . .	154
A.1.6	Torchvision . . . . .	155

<b>B Abbildungen</b>	<b>156</b>
<b>C Listings</b>	<b>175</b>
<b>D Inhaltsverzeichnis des Datenträgers</b>	<b>180</b>

# Abkürzungsverzeichnis

<b>AP</b>	Anterior-Post . . . . .	4
<b>AUC</b>	Area Under the Curve . . . . .	29
<b>AG-CNN</b>	Attention Guided Convolutional Neural Network . . . . .	68
<b>BR</b>	Binary Relevance . . . . .	35
<b>BCE</b>	Binary-Cross-Entropy . . . . .	66
<b>CAM</b>	Class Activation Mapping . . . . .	54
<b>CC</b>	Classifier Chain . . . . .	37
<b>CAD</b>	Computer-aided diagnosis . . . . .	58
<b>CLAHE</b>	Contrast Limitation Adaptive Histogram Equalization . . . . .	92
<b>CNN</b>	Convolutional Neural Network . . . . .	7
<b>DAM</b>	Deep AUC Maximization . . . . .	114
<b>FB</b>	Fusion Branch . . . . .	68
<b>FN</b>	False Negative . . . . .	25
<b>FPR</b>	False Positive Rate . . . . .	27
<b>FP</b>	False Positive . . . . .	25
<b>FC</b>	Fully-Connected . . . . .	11
<b>GAN</b>	Generative Adversarial Network . . . . .	7
<b>GB</b>	Global Branch . . . . .	68
<b>HE</b>	Histogram Equalization . . . . .	53
<b>ILSVRC</b>	ImageNet Large Scale Visual Recognition Challenge . . . . .	38
<b>IoHT</b>	Internet of Health Things . . . . .	1
<b>IoT</b>	Internet of Things . . . . .	1
<b>KI</b>	Künstliche Intelligenz . . . . .	1
<b>LP</b>	Label Powerset . . . . .	36
<b>LSR</b>	Label-Smoothing-Regularisierung . . . . .	64
<b>LB</b>	Local Branch . . . . .	68
<b>MLSMOTE</b>	Multilabel Synthetic Minority Over-sampling Technique . . . . .	94
<b>MLP</b>	Multilayer-Perceptron . . . . .	7
<b>NLP</b>	Natural Language Processing . . . . .	5
<b>OoD</b>	Out-of-Distribution . . . . .	58
<b>PWE</b>	PariWise Error . . . . .	74
<b>PA</b>	Post-Anterior . . . . .	4
<b>PCA</b>	Principle Component Analysis . . . . .	89
<b>PCAM</b>	Probabilistic Class Activation Map . . . . .	54
<b>PESG</b>	proximal epoch stochastic method . . . . .	78

<b>ROC</b>	Receiver Operating Characteristic . . . . .	29
<b>ROI</b>	Region-of-Interest	
<b>SVM</b>	Support-Vector-Machine . . . . .	74
<b>SMOTE</b>	Synthetic Minority Over-sampling Technique . . . . .	93
<b>TM</b>	Template Matching . . . . .	85
<b>TTA</b>	Test-Time-Augmentation . . . . .	6
<b>TN</b>	True Negative . . . . .	25
<b>TPR</b>	True Positive Rate . . . . .	27
<b>TP</b>	True Positive . . . . .	25
<b>UMLS</b>	Unified Medical Language System . . . . .	45

# Kapitel 1

## Einleitung

Zu Beginn dieses Kapitels wird das Thema motiviert, indem die Ausgangssituation dargestellt wird. Daraufhin wird die daraus entstehende Zielsetzung dieser Arbeit definiert und schließlich die Gliederung der Arbeit beschrieben.

### 1.1 Motivation

Bei Patienten mit Symptomen einer Lungen- oder Herzkrankheit stellt häufig das Thorax-Röntgen den ersten Schritt der medizinischen Diagnose dar [1], denn Röntgenbilder können die Grundlage für 70 - 80% aller Diagnosen bilden [2]. Obwohl zwei Drittel der Weltbevölkerung keinen gesicherten Zugang zu medizinischen Bilddaten hat [3], werden jährlich geschätzt 3.6 Milliarden Röntgenbilder aufgenommen [2, 4], wovon 2 Milliarden den Thorax betreffen [5].

Im Jahr 2019 starben in Deutschland rund 67.000 Menschen an Krankheiten des Atmungssystems [6] – weltweit sterben daran jährlich etwa 4 Millionen Menschen [7]. Aus diesen Zahlen wird klar, dass Thorax-Röntgenbilder eine sehr hohe Bedeutung in der Medizin innehaben.

Des Weiteren besitzt der medizinische Bereich viel Potenzial für technischen Fortschritt bzw. Digitalisierung. Mit *Internet of Health Things (IoHT)* bezeichnet man Internet of Things (IoT) Geräte im medizinischen Umfeld [8, S. 159f.]. Diese könnten nahezu permanent über integrierte Sensorik verschiedenste Daten erfassen und analysieren und somit medizinische Entscheidungen individualisiert unterstützen. Zusätzlich könnten sie dabei helfen, Pandemien und das Infektionsgeschehen früher und genauer zu erkennen und so einzudämmen [8, S. 164ff.]. Weitere Entwicklungsmöglichkeiten lägen in der Robotik, die beispielsweise bei Operationen, Vorsorge-Untersuchungen oder Blutabnahmen Anwendung finden könnte. Auch für die Entdeckung von Medizin oder bei der Diagnose von (seltenen) Krankheiten kann Künstliche Intelligenz (KI) unterstützend eingesetzt werden [8, S. 162f.].

Eine Vorstufe zur vollständig automatisierten Diagnose kann ein Human-In-The-Loop-System sein, bei dem Mensch und Modell im Kollektiv diagnostizieren. Patel et al. [9] untersuchten dafür ein kollaboratives Echtzeit-Zusammenarbeiten von Radiologen und

einem KI-Modell zur Klassifikation von Pneumonien auf Thorax-Röntgenbildern. Sie kamen zu dem Ergebnis, dass diese Kombination in einer Art „Schwarmintelligenz“ höhere Genauigkeiten erreicht, als die Radiologen oder das Modell alleine.

Automatische Systeme zur Diagnose von Thorax-Röntgenbildern könnten Einsatz finden bei der Erkennung schwer erkennbarer Krankheiten, der Patienten-Priorisierung oder auch in beispielsweise Entwicklungsländern, in denen Radiologen nicht immer verfügbar sind [10]. Damit bieten sie das Potenzial, die technischen Möglichkeiten der KI im Gesundheitsbereich einzubringen, um die medizinische Versorgung zu verbessern.

## 1.2 Zielsetzung

Bisherige Veröffentlichungen verbesserten zumeist durch neue Methodiken den Stand der Technik bezüglich der relevanten Metriken auf den Thorax-Röntgenbilder-Datensätzen. Stellenweise wurden auch schon innerhalb der einzelnen Ansätze Ensembling-Methoden genutzt, um die Vorhersagen zu verbessern (vgl. u.a. [11, 12, 13]). In dieser Arbeit sollen verschiedene Vorgehen des Forschungsbereichs analysiert und kombiniert sowie unter Berücksichtigung der Ensembling-Methoden untersucht werden.

## 1.3 Aufbau der Arbeit

Der Rest der vorliegenden Arbeit ist wie folgt aufgebaut.

Zunächst wird in Kapitel 2 Hintergrundwissen zum Problembereich und bestehenden Lösungen sowie technischer und theoretischer Art vermittelt. Für interessierte Leser wird im Anhang unter A.1 eine Kurzeinführung in Pytorch [14] gegeben, das in dieser Arbeit Bibliothek für die Implementierung genutzt wurde. Daraufhin wird in Kapitel 3 die Problemstellung der Multi-Label Klassifikation dargelegt, die häufig im medizinischen Kontext anzutreffen ist. In Kapitel 4 folgt eine kurze Übersicht zum Transfer Learning und der besonderen Situation bei der Anwendung auf medizinische Bilddaten, bevor in Kapitel 5 die Materialien der Arbeit vorgestellt werden.

Im darauffolgenden Abschnitt wird ein Auszug der verwandten Arbeiten vorgestellt sowie anschließend in Kapitel 7 die für die vorliegende Arbeit ausgewählten Methoden präsentiert. Auch diese Stellen einen Teil der verwandten Arbeiten dar, werden jedoch hervorgehoben, da sie zur Erreichung des Ziels dieser Arbeit implementiert wurden. Die Beschreibung der Umsetzung befindet sich in Kapitel 8.

Anschließend werden die durchgeführten Experimente sowie deren Resultate beschrieben und das Ensembling der Methoden geschildert. Daraufhin wird die Einreichung der finalen Modelle bei der CheXpert Competition [15] beschrieben, bevor in Kapitel 13 eine Zusammenfassung der Ergebnisse vorgenommen wird.

Weiterhin werden sowohl technische als auch praxisbezogene Limitationen aufgeführt. Abschließend wird die vorliegende Arbeit zusammengefasst und ein Ausblick gewagt. Der Aufbau dieser Arbeit wird in Abbildung 1.1 visualisiert.



**Abbildung 1.1:** Darstellung des Aufbaus dieser Arbeit mit einer Einordnung der Kapitel.

# Kapitel 2

## Grundlagen

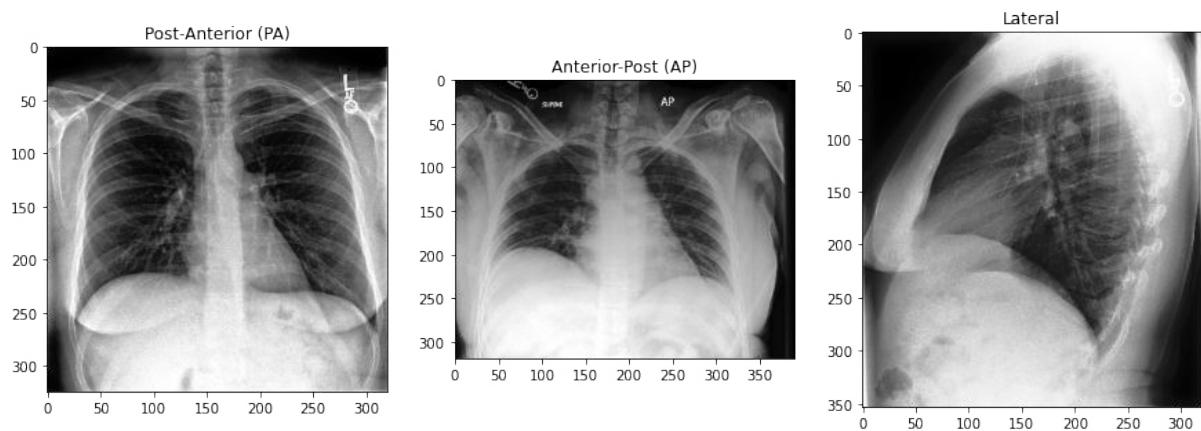
Dieses Kapitel liefert das nötige Hintergrundwissen für die vorliegende Arbeit. Es beginnt mit der Vorstellung der Datendomäne und beschreibt dann methodische und technische Grundlagen zur Bildaugmentierung, dem Einsatz von Neuronalen Netzen für die Bildklassifikation, sowie die unterschiedlichen Varianten des Ensemble Learning und wichtige Metriken zur Beurteilung der Modellgüte.

### 2.1 Thorax-Röntgenbilder

Thorax-Röntgenbilder und medizinische Bilder im Allgemeinen unterscheiden sich stark von natürlichen Bildern, die man aus dem Alltag kennt. Je nach Aufnahmeart lassen sich die meisten Thorax-Röntgenbilder in verschiedene Kategorien einteilen: frontale Aufnahmen, die weiter in Post-Anterior (PA) und Anterior-Post (AP) aufgeteilt werden, sowie laterale Bilder [10]. Für PA- und AP-Aufnahmen befindet sich das Röntgengerät hinter respektive vor dem Patienten, während laterale Bilder von der Seite, häufig von rechts, aufgenommen werden. Für AP-Bilder befinden sich die Patienten zumeist in Rückenlage und für PA-Bilder stehen sie [10]. Laterale Bilder werden häufig als zusätzliche Aufnahme zu einem PA Bild durchgeführt. In Abbildung 2.1 befindet sich ein Vergleich der verschiedenen Aufnahmepositionen.

Thorax-Röntgenbilder sind nicht trivial zu interpretieren und auch erfahrene Radiologen können bei der Betrachtung zu unterschiedlichen Diagnosen kommen, was als *Intra-Beobachter-Variabilität* bezeichnet wird [10, 16].

Damit entstehen einige Herausforderungen für den Einsatz von Deep Learning, das typischerweise große Datenmengen benötigt. Die nötige medizinische Expertise verhindert den Einsatz von Crowd-Sourced Labeling Diensten. Infolgedessen steigen auch die Kosten für das Zuweisen der Klassen zu den Bildern, da dies nur von Radiologen durchgeführt werden kann. Darüber hinaus sollten dabei idealerweise mehrere Radiologen zu Rate gezogen werden und deren Meinungsverschiedenheiten beim Labeling durch beispielsweise einen Mehrheitsentscheid gelöst werden, um die Qualität der Label zu sichern. Dies führt dazu, dass bei den großen öffentlichen Datensätzen (vgl. Kapitel 5) nur ein geringer Teil der Bilder manuell gelabelt wurde. Anhand dieser



**Abbildung 2.1:** Vergleich der Aufnahmepositionen eines Thorax-Röntgenbildes. *Links:* Post-Anterior (PA). *Mitte:* Anterior-Post (PA). *Rechts:* Lateral. Bild aus [17], Patient 184, Studien 2 und 5.

wurde dann ein Natural Language Processing (NLP)-Tool erzeugt, das die restlichen Bilder labelte.

Im medizinischen Alltag beziehen die Ärzte neben frontalen und lateralen Röntgenbildern auch andere Daten, wie Ergebnisse aus Bluttests, die Patientenhistorie und aktuelle Symptome in ihre Diagnose ein [10]. Außerdem stellt in der Praxis der Vergleich mit vorherigen Aufnahmen um den Krankheitsverlauf zu beurteilen einen wichtigen und zeitintensiven Aspekt dar [10].

## 2.2 Bestehende KI-Systeme für Thorax-Röntgenbilder

Nichtsdestotrotz sind bereits heute proprietäre Systeme zur Diagnostizierung von Thorax-Röntgenbildern im Einsatz.

Das System *qXR* [18] erhielt Ende Mai 2018 die CE-Zertifizierung, und war damit laut Herstellerangaben das erste KI-System für Thorax-Röntgenbilder, das dieses Zertifikat aufweisen kann [19]. Das System führt eine Einteilung der Bilder in Normal-Abnormal ein, erkennt bis zu 29 Befunde, zu denen es auch die Konturen der betroffenen Region liefert und ist zudem in der Lage, radiologische Berichte in Textform zu generieren. Für das Training kamen mehr als 3.7 Millionen Bilder zum Einsatz [18], die aus mehr als 40 medizinischen Einrichtungen stammen [20].

Da es sich um ein proprietäres, patentiertes System handelt, gibt das Unternehmen nur wenig Informationen zum Trainingsablauf preis. Putha et al. [20] erklären, dass sie modifizierte ResNet-Architekturen verwenden, die Bilder herunterskalieren, augmentieren und normalisieren. Außerdem wird ein Pretraining nicht auf dem ImageNet [21] durchgeführt, sondern auf der Aufgabe Thorax-Röntgenbilder von anderen Körperteilen zu unterscheiden. Darüber hinaus nutzen sie ein Majority-Ensembling.

Ein ähnliches Produkt ist *Lunit Insight CXR* [22]. Auch dieses hat die CE-Zertifizierung erhalten und soll zukünftig beispielsweise in Großbritannien, Vietnam, der Türkei, dem Mittleren Osten und Afrika eingesetzt werden [23]. Zu der Implementierung des

Systems ist wenig bekannt. Es wurde jedoch mit mehr als 3.5 Millionen Bildern trainiert und erkennt nach Herstellerangabe 10 Krankheiten mit 97-99% Genauigkeit [22]. Außerdem liefert es Konturen und Heatmaps zu den betroffenen Regionen und ist ebenfalls in der Lage einen textuellen Bericht zu generieren.

Nachdem nun der Problembereich der medizinischen Thoraxaufnahmen und KI-gestützten Diagnoseprogrammen bekannt ist, werden im Folgenden einige technische Grundlagen zur Bildverarbeitung erklärt.

## 2.3 Bildaugmentierung

In Computer Vision Projekten wird meistens sogenannte Bildaugmentierung verwendet. Damit bezeichnet man das bewusste Verändern der Bilder aus dem vorhandenen Datensatz, oft mit Zufallskomponenten, in einer Form, die das korrekte Label nicht verändert [24]. Dies kann genutzt werden, um den Datensatz künstlich zu vergrößern und / oder, um die Generalisierbarkeit des Modells zu verbessern.

Wird der Datensatz erweitert, beispielsweise indem die veränderten Bilder in einem Vorverarbeitungsschritt abgespeichert werden, spricht man auch von *Oversampling* [24]. Ansonsten wird die Bildaugmentierung auch als *Data warping* bezeichnet, wobei die beiden Begriffe nicht klar zu trennen sind.

Dies wird für gewöhnlich nur für die Trainingsdaten angewandt, es gibt aber mit der sogenannten *Test-Time-Augmentation* (TTA) auch einen Anwendungsfall für die Inferenzphase. Dabei wird für die Vorhersage der Testdaten  $n$  mal die gleiche Augmentierung wie während des Trainings angewendet und anschließend durch bilden des Mittelwertes die endgültige Vorhersage bestimmt, um diese robuster zu gestalten [24]. Dieses Vorgehen ist nur dann sinnvoll, wenn die Augmentierung Zufallskomponenten beinhaltet. Bei der TTA lässt sich somit eine Analogie zu Ensembling Methoden feststellen (vgl. Abschnitt 2.5). In Bereichen, die Echtzeit-Vorhersagen erfordern, kann der Einsatz von TTA aufgrund der erhöhten Rechendauer unpraktisch sein, sie bietet aber z.B. für den medizinischen Bereich Potenzial [24]. Hier können in der Praxis zwischen Bildaufnahme und Begutachtung durch einen Radiologen mehrere Tage vergehen [11], weswegen die  $n$ -fache Inferenzzeit, die bei Neuronalen Netzen ohnehin nur wenige Momente beträgt, nicht ins Gewicht fällt.

Die TTA kann darüber hinaus verwendet werden, um Aussagen über den Klassifizierer an sich zu treffen: Ein robustes Modell sollte bei seinen TTA-Vorhersagen nur eine geringe Varianz aufweisen [24].

Eine Bildaugmentierung sollte die Korrektheit des Labels nicht beeinträchtigen. Ob dies der Fall ist, hängt von dem verwendeten Datensatz und dessen Eigenschaften ab. Während Rotationen bei Thorax-Röntgenbildern zu keinen Änderungen an den Labels führen, könnte bei einem Datensatz zur Ziffernerkennung aus einem Bild mit der Zahl „6“ durch starke Rotation eine „9“ werden, weshalb dies als eine unsichere Bildaugmentierung bezeichnet wird (vgl. [24]).

Zu unsicheren Bildaugmentierungen für die Klassifikation von Thorax-Röntgenbildern könnten beispielsweise Änderungen an Farben und Kontrasten gezählt werden, da

Verschattungen im Lungenbereich als Krankheitsindikatoren dienen. Diese sollten also nur mit größter Vorsicht eingesetzt werden.

In den Abbildungen B.2, B.3, B.4 und B.5 werden einige wichtige Bildaugmentierungen anhand eines Beispiels vorgestellt. Die Implementierung erfolgte über die Bibliothek `torchvision` [25]. Das Ausgangsbild für alle Transformationsbeispiele befindet sich in Abbildung B.1. Neben der Möglichkeit, vorhandene Bilder zu manipulieren, könnte man auch beispielsweise durch *Generative Adversarial Networks (GANs)* Bildaugmentierung durchführen [24].

## 2.4 Neuronale Netze für die Bildklassifikation

Die Bildklassifikation ist einer der typischen Einsatzbereiche für KI. In den meisten Fällen wird dabei eine besondere Art Neuronaler Netze verwendet, sogenannte *Convolutional Neural Network* (CNN). Diese bieten für die Bildverarbeitung einige Vorteile gegenüber den klassischen vollverbundenen Multilayer-Perceptrons (MLPs).

### 2.4.1 Funktionsweise von Convolutional Neural Networks

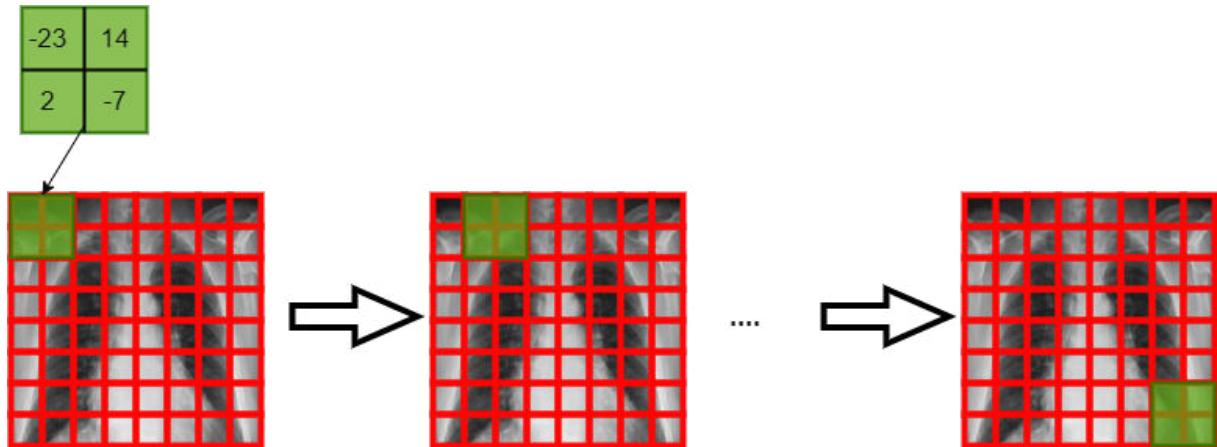
Herkömmliche MLPs sind für die Bildverarbeitung ungeeignet, da dies unter anderem eine zu große Parameterzahl erfordern würde. Daher wird für diesen Bereich meistens eine andere Netzarchitektur, sogenannte Convolutional Neural Networks (CNNs) genutzt. Diese unterscheiden sich im Wesentlichen durch zwei Operationen von MLPs: Die namensgebende *Convolution* und das *Pooling*. Außerdem ist die sogenannte *Batch-Normalization* ein wichtiger Bestandteil der meisten modernen CNN-Modelle. Diese drei Elemente werden im Folgenden kurz vorgestellt. In den darauffolgenden Abschnitten wird in zwei CNN-Architekturen eingeführt, die in der Bildklassifikation und besonders im Bereich der Thorax-Röntgenbilder weit verbreitet sind.

#### Convolution

Bei der Convolution wird schrittweise ein sogenannter *Kernel* oder *Filter* über das Eingabebild geschoben und eine *Cross-Correlation* zwischen dem aktuellen Teilbereich des Bildes und dem Kernel berechnet (vgl. Abbildung 2.2). Der Name Convolution kommt aus der Mathematik, bei der die Convolution zweier Funktionen  $f, g$  für diskrete Objekte als

$$(f * g)(i) = \sum_a f(a)g(i - a) \quad (2.1)$$

berechnet wird [26, S. 229]. Genau genommen müsste das Vorgehen einer Convolution-Schicht als *Cross-Correlation* bezeichnet werden [26, S. 231]. Diese Unterscheidung wird gewöhnlich jedoch nicht gemacht. Ein Berechnungsbeispiel befindet sich in Abbildung 2.3. Die Gewichtungen einer Convolution-Schicht bestehen lediglich in den Kernel-Elementen sowie einem Bias, der zum Ergebnis der Cross-Correlation addiert wird. Dies erklärt, warum CNNs mit deutlich weniger Parametern auskommen als MLPs. Diese Gewichtungen werden während des Trainings mittels Fehler-Backpropagation angepasst, um die Verlustfunktion zu minimieren.



**Abbildung 2.2:** Beispielhafte Darstellung der Convolution Operation mit Verschiebung des Kernels über das Eingabebild.

Input	Kernel	Output
$\begin{array}{ c c c } \hline 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 1 \\ \hline 2 & 3 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 19 & 25 \\ \hline 37 & 43 \\ \hline \end{array}$
	$\ast$	$=$

**Abbildung 2.3:** Beispiel für die Berechnung einer Convolution bzw. Cross-Correlation. Beispielsweise wird berechnet  $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$ . Quelle: [26, S. 231]

Bei der Convolution kann ein *Padding* eingesetzt werden, um Höhe und Breite des Outputs zu vergrößern. Dies wird oft genutzt, um den Output genauso groß zu gestalten wie den Input [26, S. 238f.]. Daneben gibt es noch den sogenannten *Stride*, der festlegt, um wie viele Pixel der Kernel bei jedem Schritt verschoben werden soll und so ebenfalls die Output-Dimension beeinflusst.

Mit dem Eingabebild  $n$ , Kernel  $k$ , Padding  $p$  und Stride  $s$  und jeweils  $[.]_h$  als die Dimension der Höhe und  $[.]_w$  der Breite kann die Ausgabe-Dimension einer Convolution-Operation über

$$\left\lfloor \frac{n_h - k_h + p_h + s_h}{s_h} \right\rfloor \times \left\lfloor \frac{n_w - k_w + p_w + s_w}{s_w} \right\rfloor \quad (2.2)$$

berechnet werden [26, S. 239].

Für jeden Eingabechannel wird ein Kernel benötigt, sodass sich mit den Input Channeln  $c_i$  die Kernel-Dimension als  $c_i \times k_w \times k_h$  ergibt [26, S. 242]. Meistens erhöht man innerhalb des CNNs die Kernel-Anzahl, indem für jeden Output-Channel  $c_o$  die entsprechenden Kernel angelegt werden, sodass insgesamt die Dimension  $c_o \times c_i \times k_w \times k_h$  vorliegt.

Auf diese Weise lässt sich eine Convolution mit der Kernel-Dimension  $1 \times 1$  nutzen, um die Channel-Anzahl innerhalb des Netzwerkes mittels  $c_o$  zu manipulieren [26, S. 243].

## Pooling

Das sogenannte *Pooling* basiert wie die Convolution darauf, dass ein Fenster über den Bildinhalt geschoben wird. Allerdings wird dabei, je nach Art des Poolings, entweder der Mittelwert (AveragePooling) oder das Maximum (MaxPooling) des Fensters als Ausgabewert genutzt [26, S. 245ff.]. Das Pooling fügt dem CNN somit eine gewisse Translations-Invarianz hinzu und wird häufig mit einem Stride  $s > 1$  angewendet, um ein Downsampling zu erreichen. Weder das Average noch das Maximum Pooling besitzen trainierbare Parameter.

Convolution und Pooling führen dazu, dass die tieferen Schichten ein größeres rezeptives Feld besitzen, also gesamte Menge an Eingabepixeln, die bei den Berechnungen berücksichtigt wurde, mit zunehmender Schichtanzahl steigt. Dies führt dazu, dass die früheren Schichten eines CNNs einfache Features wie beispielsweise Kanten lernen, während spätere Schichten zusammengesetzte Features, z.B. Gesichtsformen lernen [27].

## Batch-Normalization

Eine Operation, die unter anderem auch für CNNs vorteilhaft ist, ist die *Batch-Normalization* [28]. Dabei wird für einen Mini-Batch  $\mathcal{B} = \{x_1, \dots, x_m\}$  der arithm. Mittelwert  $\bar{x}_{\mathcal{B}}$  und die empirische Varianz  $s_{\mathcal{B}}^2$  des Batches über

$$\bar{x}_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^m x_i, \quad s_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \bar{x}_{\mathcal{B}})^2 \quad (2.3)$$

berechnet und anschließend die Datenpunkte des Batches normalisiert mit

$$\hat{x}_i = \frac{x_i - \bar{x}_{\mathcal{B}}}{\sqrt{s_{\mathcal{B}}^2 + \epsilon}}, \quad (2.4)$$

wobei  $\epsilon$  eine Konstante ist, die lediglich zur numerischen Stabilität dient [28]. Die Batch-Normalization führt dazu, dass Gradienten nicht mehr explodieren oder verschwinden, beschleunigt das Training, erlaubt höhere Lernraten und bringt einen Regularisierungseffekt ein [28]. Durch ein Ensembling von Batch-Normalisierten Modellen konnten Ioffe et al. [28] den Stand der Technik des Top-5-Fehlers auf den Testdaten des ImageNet-Datensatzes [21] verbessern.

### 2.4.2 ResNet

Nach dem Durchbruch von CNNs zur Bildverarbeitung stellte man schnell fest, dass tiefere Netzwerke, also solche mit mehr Schichten, zu besseren Ergebnissen führen [29]. Anfangs war es möglich, Netzwerke mit etwa 16 bis 30 Schichten zu trainieren. Allerdings zeigten He et al. [29], dass bei tieferen Netzwerken die Genauigkeit ab einem gewissen Trainingszeitpunkt rapide abfällt und der Trainingsfehler von tieferen Netzen *höher* ist, als bei vergleichbaren flacheren Architekturen. Ein höherer Trainingsfehler bedeutet im Allgemeinen auch einen höheren Testfehler und das Modell ist somit schlichtweg schlechter (siehe Abbildung 2.4).

He et al. [29] argumentieren, dass eine tiefere Version einer Netzarchitektur keinen größeren Trainingsfehler produzieren sollte als die flachere, da die zusätzlichen Schichten auch als reine Identitätsfunktion  $f(x) \rightarrow x$  fungieren könnten.

Da dies aber nicht der Fall ist, stellen sie ihre Methode vor, die es erlaubt, tiefere Netze erfolgreich zu trainieren. Dafür werden *shortcut connections* verwendet, die als Identitätsfunktion den Schicht-Input an einer (oder mehreren) Schichten vorbei erhalten und anschließend auf den Output addieren (siehe Abbildung 2.5).

Dadurch ändert sich die vom Netz zu erlernende Funktion  $\mathcal{H}(x)$  auf  $\mathcal{F}(x) := \mathcal{H}(x) - x$ . Da die Identitätsfunktion keine Parameter beinhaltet, ändert sich der Rechenaufwand nur minimal.

Die Dimensionen innerhalb eines CNNs sind allerdings für gewöhnlich nicht konstant, was zu Problemen beim Addieren nach der shortcut connection führen würde. Für den Fall, dass die Dimensionen am Start- und Endpunkt einer shortcut connection nicht identisch sind, verglichen die Autoren daher zwei Strategien.

Entweder können fehlende Dimensionen mit einem Nullen-Padding aufgefüllt werden, oder die Dimensionen durch eine Convolution mit Filtergröße  $1 \times 1$  über die Filteranzahl angepasst werden, was die Autoren als *projection shortcuts* bezeichnen. Die so entstehenden Netzwerke werden als Residual Networks, kurz *ResNets*, bezeichnet. In Abbildung 2.6 wird ein Vergleich zwischen zwei ResNets und den entsprechenden CNNs ohne shortcut connections auf dem ImageNet-Datensatz dargestellt. Es lässt sich erkennen, dass ohne die shortcut connections die höhere Schichtenanzahl nicht zu einer Reduzierung der Fehler führt, das tiefere ResNet hingegen einen geringeren Trainings- und Validierungsfehler aufweist.

Die Autoren gehen nicht davon aus, dass die Defizite der Netzwerke ohne shortcut connections durch verschwindende Gradienten ausgelöst werden, da dies durch den Einsatz der Batch-Normalization verhindert wird.

Des Weiteren führten die Autoren ein Experiment durch, bei dem sie verglichen, wie sich Nullen-Padding im Vergleich zu  $1 \times 1$  Convolutions und dem pauschalen Einsatz von projection shortcuts statt parameterfreien Identitäts-shortcut-connections auswirkt. Dies zeigte, dass die projection shortcuts keinen relevanten Vorteil bringen, weswegen die Autoren zugunsten der geringeren Parameteranzahl und Speicherkomplexität im Weiteren darauf verzichtet haben.

He et al. [29] stellen ResNets in fünf verschiedenen Tiefen vor: Mit je 18, 34, 50, 101 und 152 Schichten. Bei den ersten beiden besteht ein Architektur-Baustein aus zwei  $3 \times 3$  Convolutions während bei den drei tieferen Architekturen ein sogenannter *bottleneck* Block aus einer Abfolge von einer  $1 \times 1$ ,  $3 \times 3$  und  $1 \times 1$  Convolution besteht (siehe Abbildung 2.7). Der bottleneck Block nutzt die  $1 \times 1$  Convolutions um über die Filteranzahl die Dimension vor der  $3 \times 3$  Convolution zu reduzieren und sie anschließend wieder auf den Ursprungswert zu erhöhen.

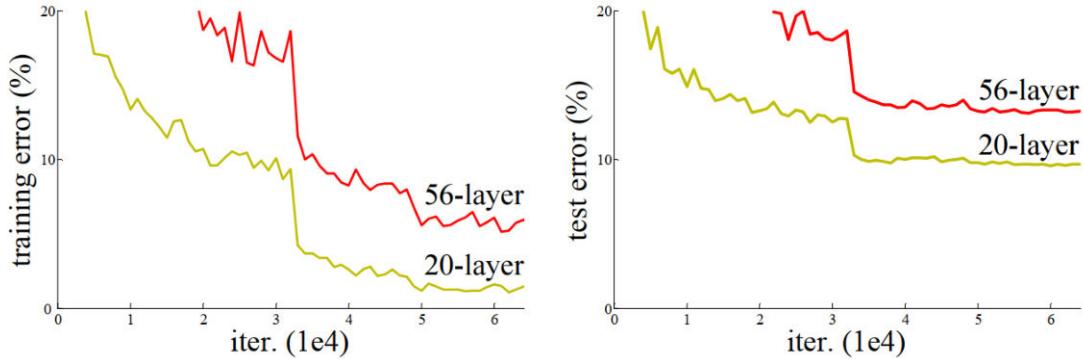
Für das ResNet-34 bedeutet dies, dass  $(34 - 2) / 2 = 16$  Blöcke verwendet werden. Die restlichen beiden Schichten bestehen aus der Eingabe-Convolutional-Schicht und der Fully-Connected (FC)-Schicht, die für den Output zuständig ist. Dahingehen verwendet das ResNet-50 16 bottleneck Blöcke was mit der Ein- und Ausgabeschicht genau  $(3 \times 16) + 2 = 50$  Schichten ergibt. Die tieferen Netze werden entsprechend mit mehr bottleneck Blöcken konstruiert.

In diesen Architekturen kommt das Pooling nur nach der Input- und vor der Output-Schicht vor. Genauer gesagt wird nach der ersten Convolution-Schicht ein MaxPooling und vor der FC-Schicht ein AveragePooling angewendet. Typischerweise wird das Pooling verwendet, um innerhalb des Netzwerks ein Downsampling durchzuführen. In der ResNet-Architektur wird dies durch eine Convolution mit einem Stride von  $s = 2$  erreicht. Die Autoren teilen alle Architekturen in fünf größere Convolution-Blöcke ein, wobei der erste nur aus der Eingabeschicht besteht. Die Übersicht über die Architekturen befindet sich in Abbildung 2.8. Es sollte darauf hingewiesen werden, dass selbst das größte ResNet mit seinen 152 Schichten *weniger* Berechnungsschritte erfordert als ein VGG-16, das 16 Schichten beinhaltet [29]. Dies lässt sich auf die geringere Anzahl an FC-Schichten und Neuronen in diesen zurückführen.

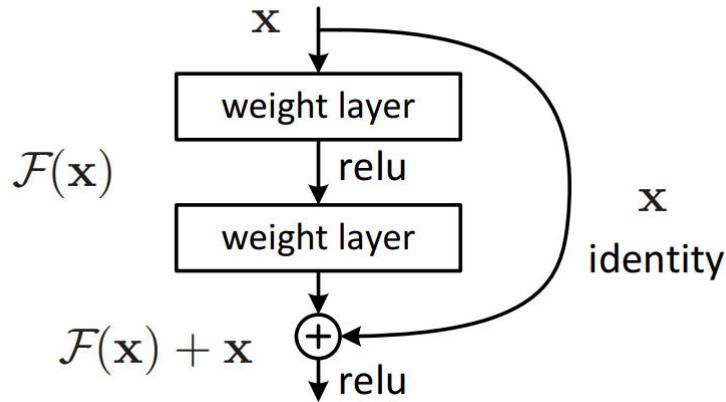
Die Effizienz dieser Architekturen wird auch daran klar, dass die Autoren mit ihnen jeweils die ersten Plätze bei der ILSVRC 2015 (vgl. Kapitel 4.1) in den Kategorien Bildklassifizierung, ImageNet Detektion und ImageNet Lokalisierung gewannen, sowie auch beim COCO Wettbewerb in den Konkurrenzen Detektion und Segmentierung [29].

### 2.4.3 DenseNet

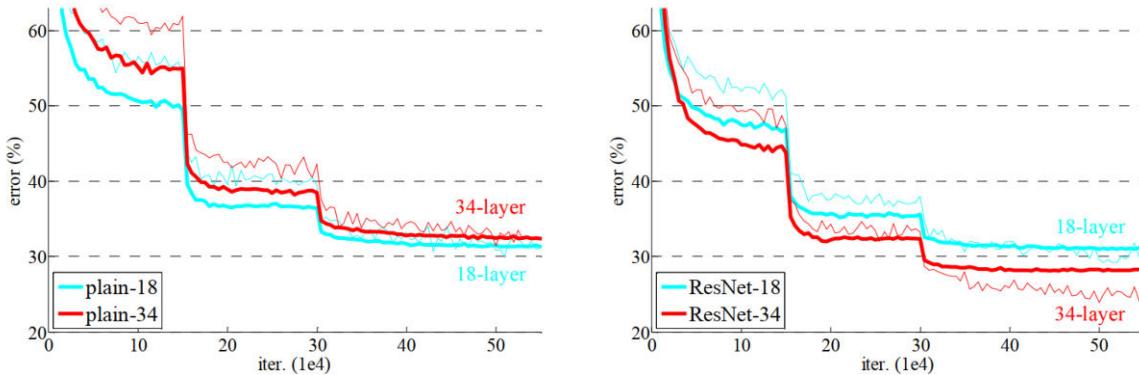
Die sogenannten DenseNets wurden von Huang et al. [30] vorgestellt und basieren auf einer ähnlichen Grundidee wie die ResNets: Die Outputs von vorherigen Schichten werden mit dem Input der aktuellen Schicht kombiniert. Die wichtigsten Unterschiede zwischen DenseNets und ResNets sind, dass DenseNets nicht nur den Output einer, sondern *aller* vorherigen Schichten für die Kombination nutzt und diese nicht als



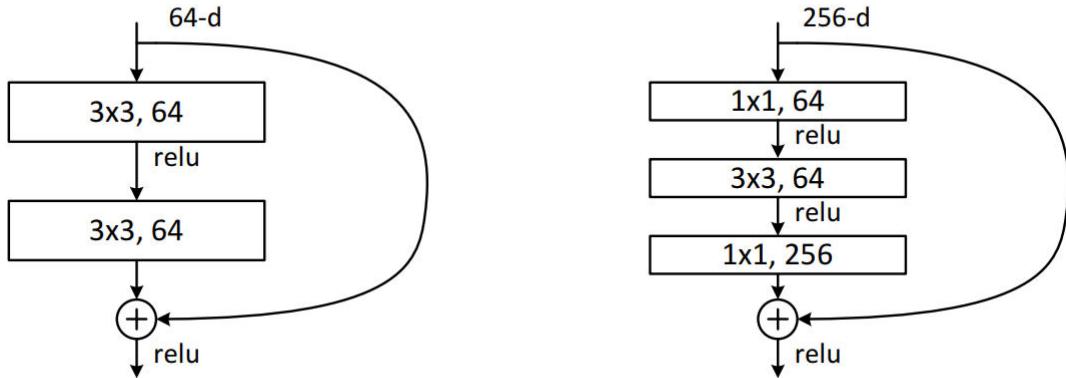
**Abbildung 2.4:** Vergleich des Trainings- (links) und Testfehlers zweier Netze mit 20 respektive 56 Schichten auf dem CIFAR-10-Datensatz. Das flachere Netzwerk performt sowohl bei den Trainings- als auch den Testdaten besser. Quelle: [29].



**Abbildung 2.5:** Darstellung einer shortcut connection. Quelle: [29].



**Abbildung 2.6:** Vergleich zwischen ResNets (rechts) und CNNs ohne shortcut connections mit 18 und 34 Schichten auf dem ImageNet-Datensatz. Die dünnen Kurven stellen den Trainings- und die dicken Kurven den Validierungsfehler dar. Durch die shortcut connections führt die Tiefe der ResNets zu geringeren Fehlern. Quelle: [29].



**Abbildung 2.7:** Darstellung eines Blocks für ResNet-18 und ResNet-34 (links) sowie eines bottleneck Blocks für die tieferen ResNets. Quelle: [29].

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

**Abbildung 2.8:** Übersicht über die ResNet-Architekturen. Für conv3\_1, conv4\_1 und conv5\_1 wird die Convolution mit einem Stride von  $s = 2$  ausgeführt um ein Downsampling zu erreichen. Quelle: [29].

Layers	Output Size	DenseNet-121( $k = 32$ )	DenseNet-169( $k = 32$ )	DenseNet-201( $k = 32$ )	DenseNet-161( $k = 48$ )
Convolution	$112 \times 112$		$7 \times 7$ conv, stride 2		
Pooling	$56 \times 56$		$3 \times 3$ max pool, stride 2		
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$			$1 \times 1$ conv	
	$28 \times 28$			$2 \times 2$ average pool, stride 2	
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$			$1 \times 1$ conv	
	$14 \times 14$			$2 \times 2$ average pool, stride 2	
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	$14 \times 14$			$1 \times 1$ conv	
	$7 \times 7$			$2 \times 2$ average pool, stride 2	
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	$1 \times 1$			$7 \times 7$ global average pool	1000D fully-connected, softmax

**Abbildung 2.9:** Übersicht über die DenseNet-Architekturen wie sie von Huang et al. [30] vorgestellt wurden. Quelle: [30].

Summation, sondern als Konkatenierung erfolgt. Für ein Netz mit  $L$  Schichten lässt sich die Anzahl der Verbindungen mit der Gauß'schen Summenformel berechnen als  $\frac{L(L+1)}{2}$ .

Die Autoren begründen dieses Design mit der Wiederverwendung bereits in vorherigen Schichten gelernten Features. Huang et al. [30] präsentieren vier DenseNets mit 121, 169, 201 und 161 Schichten.

Ähnlich wie bei den ResNets definieren die Autoren die DenseNets über sogenannte *Dense Blocks*, die aus Batch-Normalisierung (BN), ReLU und Convolutions mit Filtergrößen  $1 \times 1$  und  $3 \times 3$  bestehen. Der erste Dense Block des DenseNet-121 beinhaltet beispielsweise sechs mal die Abfolge BN-ReLU- $1 \times 1$  Conv-BN-ReLU- $3 \times 3$ Conv. Die  $1 \times 1$  Convolution realisiert die bottleneck-Strategie, die auch die ResNets nutzen.

Zwischen zwei Dense Blocks bringen die Autoren sogenannte *Transitional Layer* ein, die aus BN-ReLU- $1 \times 1$  Convolution und einem AveragePooling bestehen. Durch diese wird das Downsampling erreicht.

Die Autoren benennen zwei Hyperparameter, über die sich die Architektur eines DenseNet steuern lässt: Die *Growth rate* und den *Compression factor*.

Die Growth rate  $k$  entspricht der Anzahl der Output-Filter bzw. Feature-Maps der  $3 \times 3$  Convolution Schichten. Somit bestimmt  $k$  die Breite des Netzwerkes. Die Anzahl der Eingabefilter einer  $1 \times 1$  Conv als  $l$ -te Schicht lässt sich mit der Anzahl der Filter in der Input-Schicht  $k_0$  bestimmen als  $k_0 + k \times (l - 1)$  [30].

Der Compression factor  $\theta$  kann hingegen verwendet werden, um die Anzahl der Feature-Maps, die ein Transitional Layer ausgibt, zu reduzieren.

Sei  $m$  die Anzahl der Feature-Maps eines Dense-Blocks, dann generiert das darauffolgende Transitional Layer  $\lfloor \theta m \rfloor$  Feature-Maps mit  $0 < \theta \leq 1$  [30].

Eine detaillierte Übersicht über die Architekturen befindet sich in Abbildung 2.9. Mit diesen Architekturen gelang es den Autoren bei geringerer Parameterzahl kleinere Fehler als ResNets zu erreichen.

## 2.5 Ensemble Learning

Unter einem sogenannten *Ensemble* bzw. *Ensemble Learning* versteht man zunächst einen Zusammenschluss mehrerer Vorhersagemodelle. Dabei werden die einzelnen Modellausgaben auf verschiedene Weise kombiniert, um die endgültige Vorhersage zu treffen [31, S. 1]. Ensembles können sowohl für Regressions- als auch Klassifikationsprobleme angewendet werden.

Goodfellow et al. [32, S. 256] zeigen, dass Ensembling im Allgemeinen zu einem niedrigeren Generalisierungsfehler führt:

Sei  $k$  die Anzahl der Vorhersagemodelle die das Ensemble bilden,  $\epsilon_i$  der Fehler eines Modells auf dem  $i$ -ten Datenpunkt. Unter der Annahme, dass die Fehler  $\epsilon_i$  normalverteilt und unabhängig sind, sei die Varianz  $v := \mathbb{E}[\epsilon_i^2]$  und die Kovarianz zweier Fehler  $c := \mathbb{E}[\epsilon_i \epsilon_j]$ . Der Fehler des Ensembles beträgt als Mittelwert über die Summe der Einzelfehler  $\frac{1}{k} \sum_i \epsilon_i$ . Der erwartete quadrierte Fehler (engl. *Expected Squared Error*) des Ensembles lässt sich dann nach Goodfellow et al. [32, S. 256] schreiben als

$$\mathbb{E} \left[ \left( \frac{1}{k} \sum_i \epsilon_i \right)^2 \right] = \frac{1}{k^2} \mathbb{E} \left[ \sum_i \left( \epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j \right) \right] = \frac{1}{k} v + \frac{k-1}{k} c . \quad (2.5)$$

Der Ensemble-Fehler hängt also von den Fehlern der einzelnen Modelle ab. Die Art dieses Zusammenhangs wird durch betrachten zweier Extremfälle klar:

Angenommen, jedes Modell des Ensembles würde die exakt gleiche Vorhersage treffen, beispielsweise da sie naiverweise perfekte Kopien voneinander sind. In diesem Fall wären auch die Fehler  $\epsilon_i$  gleich. Damit wäre die Kovarianz zwischen den Fehlern gleich der Varianz der Fehler:  $c = v$  [32, S. 256]. Dies lässt sich einfach sehen wenn man die allgemeinen Formeln für die empirische Varianz  $s^2(x)$  eines Datensatzes der Länge  $n$  mit den Datenpunkten  $x_i$  wobei  $i \in \{1, \dots, n\}$  und für die empirische Kovarianz  $s_{xy}$  zwischen  $x_i$  und einer weiteren Variable  $y_i$  betrachtet. Die empirische Varianz berechnet sich als

$$s^2 := \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 , \quad (2.6)$$

wobei  $\bar{x}$  das arithmetische Mittel aller  $x_i$  ist [33, S. 81]. Und die empirische Kovarianz kann mit dem arithmetischen Mittel  $\bar{y}$  aller  $y_i$  über

$$s_{xy} := \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (2.7)$$

berechnet werden [33, S. 146].

In dem aktuellen Gedankengang sind nun aber die  $\epsilon_i$  gleich, womit sich die empirische Kovarianz  $s_{xy}$  zwischen den Fehlern als  $s_{\epsilon_i \epsilon_i}$  berechnet und sich somit zur Varianz reduziert:

$$s_{\epsilon_i \epsilon_i} := \frac{1}{n} \sum_{i=1}^n (\epsilon_i - \bar{\epsilon}_i) (\epsilon_i - \bar{\epsilon}_i) = \frac{1}{n} \sum_{i=1}^n (\epsilon_i - \bar{\epsilon}_i)^2 = s^2. \quad (2.8)$$

Die Gleichung (2.5) lässt sich unter Berücksichtigung von Formel (2.8) also auch als

$$\mathbb{E} \left[ \left( \frac{1}{k} \sum_i \epsilon_i \right)^2 \right] = \frac{1}{k} v + \frac{k-1}{k} v = \frac{k}{k} v = v \quad (2.9)$$

schreiben. In diesem Fall wäre der Fehler des Ensembles gleich  $v$  und damit gleich dem Fehler eines einzelnen (beliebigen) Modells aus dem Ensemble. Dies würde jegliches Ensembling überflüssig machen. Daraus lässt sich bereits schließen, dass die Ensemble-Modelle unterschiedlich sein sollten, um so nicht-identische Fehler zu produzieren.

Betrachtet man das andere Extrem, wären die  $\epsilon_i$  also gänzlich unkorreliert und die Fehler  $\epsilon_i, \epsilon_j$  der einzelnen Modelle somit alle unabhängig und unterschiedlich, wäre die Kovarianz als Maß der Korreliertheit  $\mathbb{E}[\epsilon_i \epsilon_j] = c = 0$  [32, S. 256]. In diesem Fall lässt sich Gleichung (2.5) als

$$\mathbb{E} \left[ \left( \frac{1}{k} \sum_i \epsilon_i \right)^2 \right] = \frac{1}{k} v + \frac{k-1}{k} 0 = \frac{1}{k} v \quad (2.10)$$

kürzen. Mit steigender Anzahl der Modelle  $k$  würde also auch der Ensemble-Fehler sinken. Je mehr gänzlich unkorrelierte Modelle man also in ein Ensemble überführen würde, desto geringer wäre der Fehler.

Dies verdeutlicht, dass es wünschenswert ist, diversifizierte Modelle zu einem Ensemble zu kombinieren, denn je unabhängiger voneinander bzw. unterschiedlicher die Einzelfehler sind, desto besser wird das Ensemble.

Selbst im ungünstigsten Szenario ( $c = v$ ) führt das Ensembling im Durchschnitt immerhin nicht zu einer Verschlechterung der Vorhersagen, jedoch steigt der Ressourcenbedarf unnötig.

Kumar et al. [31, S. 2] teilen die Ensembling Methoden grundlegend in drei Kategorien ein: Mischung der Trainingsdaten, Mischung der Kombinationen und Mischung der Modelle.

### 2.5.1 Mischung der Trainingsdaten

Dieser Ansatz besteht darin, die Ensemble-Teile mit zufälligen, unterschiedlichen Teilmengen der Trainingsdaten zu trainieren [31, S. 12]. Die verwendeten Modelle sind dabei Instanzen des gleichen Modell-Typs, beispielsweise einer Logistischen Regression.

Die intuitive Begründung dieses Vorgehens besteht darin, dass die Teilmengen jeweils einer anderen Verteilungen unterliegen als der ursprüngliche Trainingsdatensatz und

somit zu diversifizierten Einzelmodellen führen, die dann, wie oben beschrieben, gewinnbringend kombiniert werden können [31, S. 12]. Dies kann auch ein großer Vorteil sein, wenn die Verteilung der Trainingsdaten nicht die Verteilung der Realität abbildet.

Das sogenannte *Bagging* von engl. **Bootstrap Aggregating** ist eine Option zur Umsetzung des Prinzips Mischung der Trainingsdaten. Dabei werden aus dem Trainingsdatensatz für  $n$  Modelle  $n$  Teilmengen zufällig mit Zurücklegen gezogen („*Bootstrapping*“) [31, S. 20]. Anschließend werden die einzelnen Modelle auf jeweils einer der Teilmengen trainiert. Für die Vorhersage werden die Ausgaben der Ensemble-Teile kombiniert („*Aggregating*“) [31, S. 21].

Auch die  $k$ -fache Kreuzvalidierung, die häufig zum Hyperparameter-Tuning und zur besseren Einschätzung der Generalisierbarkeit eines Modells verwendet wird, kann für die Bildung eines Ensembles genutzt werden [31, S. 25]. Dabei wird der Datensatz in  $k$  Subsets aufgeteilt und das Modell  $k$  mal trainiert. Dafür werden immer  $k - 1$  Subsets als Trainingsdaten und ein Subset als Validierungsdatensatz verwendet. Dies führt dazu, dass verschiedene Distributionen als Validierungsdaten zum Einsatz kommen, wodurch die Generalisierungsfähigkeit besser eingeschätzt werden kann. Gleichzeitig werden die  $k$  Modelle aber auch alle mit leicht unterschiedlichen Trainingsdaten trainiert, weswegen ein derartiges Ensembling mittels  $k$ -facher Kreuzvalidierung der Mischung der Trainingsdaten zuzuordnen ist [31, S. 25].

Goodfellow et al. [32, S. 257f.] weisen darauf hin, dass Neuronale Netze durch verschiedene probabilistische Elemente wie die zufällige Parameterinitialisierung, unterschiedliche Hyperparameter und zufälliges mischen der Mini-Batches auch beim Training auf dem gleichen Datensatz unterschiedlich genug sind, um partiell unabhängige Fehler zu erzeugen.

Die im Bereich der Künstlichen Visuellen Intelligenz häufig verwendete Bildaugmentierung beinhaltet üblicherweise ebenfalls probabilistische Elemente und verstärkt somit diesen Effekt.

Gleichzeitig möchte man für gewöhnlich alle verfügbaren Daten nutzen um möglichst gute Modelle zu erzeugen. Daher ist es bei der Verwendung von Deep Learning Modellen üblich, keine Teilmengen zu bilden, sondern jedes Modell auf allen Trainingsdaten zu trainieren [32, S. 257f.].

## 2.5.2 Mischung der Modelle

Bei der Mischung der Modelle wird das Ensemble aus Modellen verschiedenen Typs gebildet, die mit dem gleichen Datensatz trainiert wurden [31, S. 31]. Dabei unterscheidet man zwischen Voting Ensembles, Hyperparameter Tuning Ensembles, Horizontal Voting Ensembles und Snapshot Ensembles.

Voting Ensembles lassen sich weiter in Hard und Soft Voting deklinieren [31, S. 33ff.]. Bei einem Hard Voting Ensemble wird das Prinzip eines Mehrheitsentscheids angewendet. Für ein Klassifizierungsproblem würde dies bedeuten, dass die am häufigsten von den Ensemble-Teilen vorhergesagte Klasse als Ausgabe des Ensembles verwendet wird.

Beim Soft Voting wird hingegen ein (gewichteter) Mittelwert über die Ausgaben der Ensemble-Teile gebildet, der dann die Ensemble-Vorhersage bildet. Über die Gewichtung ist es möglich, einzelne Klassifizierer stärker in die Entscheidung einfließen zu lassen, beispielsweise wenn bekannt ist, dass ein bestimmtes Modell besonders gute Prognosen liefert. Je unsicherer die Modelle bei ihren Vorhersagen sind, desto größer kann der Unterschied zwischen Soft und Hard Voting sein.

Beim Hyperparameter Tuning Ansatz besteht das Ensemble aus Instanzen des gleichen Modell-Typs allerdings mit signifikant unterschiedlichen Hyperparametern [31, S. 38]. Beispielsweise könnte man ein Ensemble aus k-Nearest-Neighbor-Klassifizierern konstruieren und dabei die Anzahl der verwendeten Nachbarn variieren.

Horizontal Voting Ensembles bieten sich besonders für Neuronale Netze an, die zumeist mehrere Trainingsiterationen auf einem Datensatz (sogenannte Epochen) durchlaufen. Dabei schwankt die Güte des Modells zwischen den Epochen und konvergiert typischerweise ab einem gewissen Zeitpunkt. Die Idee des Horizontal Votings besteht darin, das Ensemble aus dem trainierten Modell nach verschiedenen Epochenzahlen zu bilden [31, S. 40f.]. Beispielsweise könnte alle fünf Epochen, nachdem mindestens 25 Epochen durchlaufen sind, ein Modell-Checkpoint abgespeichert werden. Auch da es im Allgemeinen nicht trivial ist, die beste Epochenzahl festzulegen, kann das Horizontal Voting eine sinnvolle Methodik sein. Anschließend kann sowohl ein Hard als auch Soft Voting angewendet werden.

Das grundlegende Vorgehen beim Snapshot Ensembling ist ähnlich, allerdings wird dieses in Kombination mit zyklischen Lernraten verwendet. Beim Einsatz von zyklischen Lernraten wird die initiale Lernrate über eine gewisse Epochendauer reduziert und dann auf den ursprünglichen Wert zurückgesetzt. Dies wird während des Trainings mehrfach wiederholt. Für ein Snapshot Ensemble nimmt man dann die letzten Modellzustände vor dem Zurücksetzen der Lernrate [31, S. 46 f.]. So soll erreicht werden, dass die Ensemble-Teile aus unterschiedlichen lokalen Minima entstanden sind. Denn durch das Zurücksetzen wird die Lernrate erhöht und das Modell könnte sich aus dem vorherigen lokalen Minima „befreien“.

### 2.5.3 Mischung der Kombinationen

Die Ansätze der Mischung der Kombinationen lassen sich in Boosting und Stacking einordnen.

Beim Boosting wird ein Modell iterativ verbessert, indem der Fokus besonders auf dessen Schwächen in der vorherigen Iteration gelegt wird.

Eine Boosting-Methode ist der sogenannte *AdaBoost*. Dabei wird in jeder Iteration ein neues Modell trainiert, für dessen Eingabe die Gewichte der falsch klassifizierten Objekte erhöht werden und somit die Wahrscheinlichkeit, dass sie in dieser Iteration richtig klassifiziert werden [31, S. 50f.].

Im Gegensatz dazu werden beim GradientBoosting als Eingabe für das Modell der aktuellen Iteration die Residuen der vorherigen Iteration verwendet [31, S. 52]. Die Residuen sind die Abweichungen zwischen dem Zielwert und der Vorhersage eines Modells. Das heißt, dass jedes Modell im Ensemble die Fehler des vorherigen Modells

additiv ausgleichen soll. Für die Ensemble-Vorhersage werden dementsprechend die Ausgaben der einzelnen Modelle addiert.

Während AdaBoost robust gegen Overfitting ist, sollten beim GradientBoosting Maßnahmen dagegen ergriffen werden, beispielsweise mittels Regularisierung [34]. Beispielsweise ist XGBoost [35] ein Algorithmus, der das GradientBoosting unter anderem auf diese Weise verbessert [31, S. 55].

Beim Stacking geht es darum, die bestmögliche Funktion zur Kombination der Ausgaben der Ensemble-Teile zu finden, statt beispielsweise nur ein Voting durchzuführen. Die Vorhersagen der einzelnen Modelle werden also als Eingabe für ein eigenes, „Meta-Modell“ verwendet [31, S. 57f.].

In Abbildung 2.10 findet sich eine Übersicht über die verschiedenen Ensembling Methoden.

#### 2.5.4 Committee Machines

In [36, S. 373] führt der Autor den Begriff der Committee Machines ein und versteht darunter die Kombination mehrerer Modelle, die jeweils „Experten“ für eine bestimmte Aufgabe sind. Die Committee Machines unterteilt er in vier Arten, die sich in zwei Kategorien einteilen ließen [36, S. 373]:

1. Statische Strukturen: Der Input wird bei der Kombination nicht direkt berücksichtigt.
  - Ensemble Averaging: Lineare Kombination der Experten-Outputs.
  - Boosting: Aus einem schwachen wird ein Algorithmus beliebiger Genauigkeit erzeugt.
2. Dynamische Strukturen: Der Input wird bei der Kombination direkt berücksichtigt.
  - Mischung der Experten: Nicht-Lineare Kombination der Experten-Outputs.
  - Hierarchische Mischung der Experten: Nicht-Lineare hierarchische Kombination der Experten-Outputs.

Bereits aus den Kurzbeschreibungen wird klar, dass die Prinzipien der Committee Machines in dieser Arbeit bereits als Ensembling Techniken unter anderem Namen eingeführt wurden, siehe Abschnitte 2.5.1, 2.5.2 und 2.5.3. Die Ensemble Averaging Committee Machine entspricht dem Voting Ensemble, der Begriff Boosting wird in beiden Referenzwerken identisch verwendet und die Mischung der Experten kann als Stacking gesehen werden. Einzig die Hierarchische Mischung der Experten findet nicht direkt eine Entsprechung in den bisherigen Beschreibungen, da sie aber eine Sonderform der Mischung der Experten ist, und das Stacking hierarchische Meta-Lerner nicht ausschließt, könnte man es ebenfalls dem Stacking zuordnen.

In [36, S. 375ff.] zeigt der Autor, dass es beim Ensemble Averaging vorteilhaft sein kann, dass die einzelnen Experten overfitten, um den Fehler der Committee Machine zu

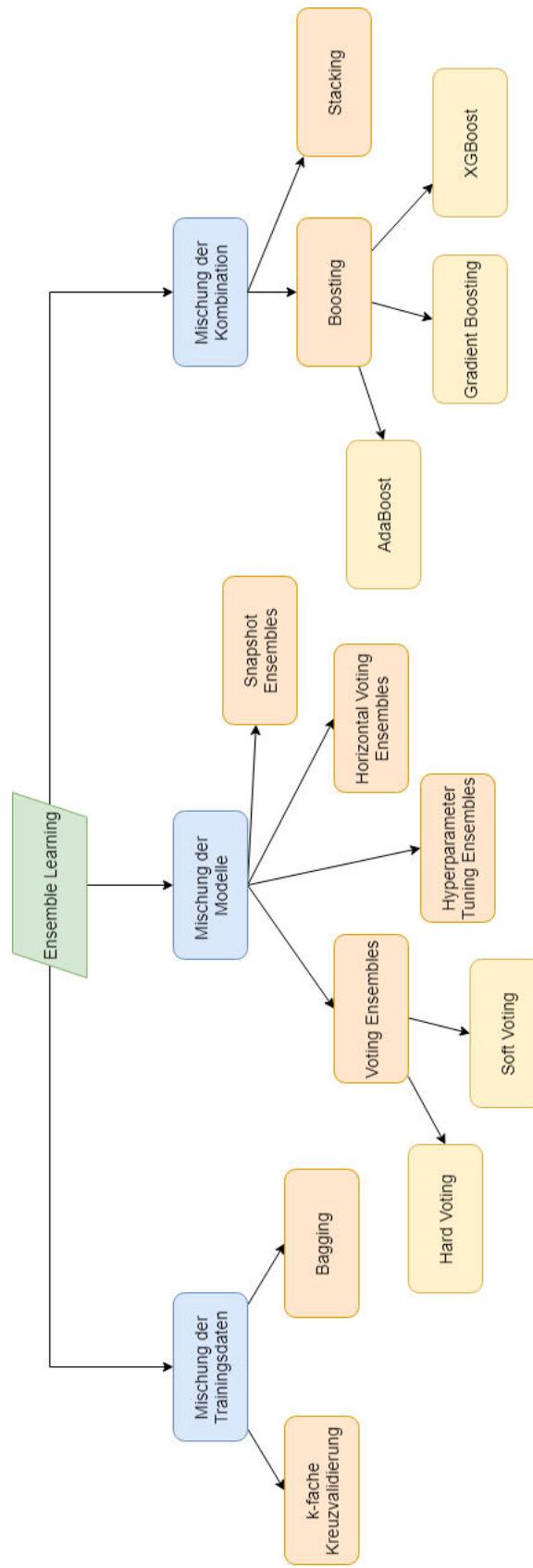


Abbildung 2.10: Übersicht der verschiedenen Ensemble Möglichkeiten und deren Kategorisierung.

reduzieren: Sei  $\mathbf{x}$  ein Input-Vektor aus den Inputs  $\mathbf{X}$  und sei analog für die Outputs  $\mathbf{y}$  eine Realisierung aus  $\mathbf{Y}$ . Zudem sei  $\mathcal{D}$  der Raum, der die Verteilung aller Trainingsdaten und aller initialen Bedingungen beinhaltet und  $F(\mathbf{x})$  die Netzwerkfunktion, die aus  $\mathbf{x}$  ein  $\mathbf{y}$  vorhersagt. Die Herleitung beginnt zunächst mit der Aufteilung des Mean-Squared Errors in Bias und Varianz, wie es aus dem *bias/variance dilemma* [36, S. 109ff.] bekannt ist,

$$\mathbb{E}_{\mathcal{D}} \left[ (F(\mathbf{x}) - \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}])^2 \right] = B_{\mathcal{D}}(F(\mathbf{x})) + V_{\mathcal{D}}(F(\mathbf{x})) , \quad (2.11)$$

mit dem quadrierten Bias  $B_{\mathcal{D}}(F(\mathbf{x}))$  als

$$B_{\mathcal{D}}(F(\mathbf{x})) = (\mathbb{E}_{\mathcal{D}}[F(\mathbf{x})] - \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}])^2 , \quad (2.12)$$

und der Varianz  $V_{\mathcal{D}}(F(\mathbf{x}))$  als

$$V_{\mathcal{D}}(F(\mathbf{x})) = \mathbb{E}_{\mathcal{D}} \left[ (F(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}[F(\mathbf{x})])^2 \right] \quad (2.13)$$

[36, S. 375].

Nun wird die Ausgabe der Committee Machine, die als Mittelwert gebildet wird, bezeichnet als  $F_I(\mathbf{x})$  und  $\theta$  bezeichnet den Raum aller initialer Bedingungen. Damit lässt sich analog zu Gleichung (2.11) der Fehler der Committee Machine zerlegen

$$\mathbb{E}_{\theta} \left[ (F_I(\mathbf{x}) - \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}])^2 \right] = B_{\theta}(F(\mathbf{x})) + V_{\theta}(F(\mathbf{x})) , \quad (2.14)$$

wodurch die Zusammensetzung aus Bias und Varianz klar wird [36, S. 376].

Anschließend wird der Raum  $\mathcal{D}$  betrachtet als Produkt zwischen den Startbedingungen  $\theta$  und dem Restraum  $\mathcal{D}'$  und damit erneut analog zu Gleichung (2.11) formuliert

$$\mathbb{E}_{\mathcal{D}'} \left[ (F(\mathbf{x}) - \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}])^2 \right] = B_{\mathcal{D}'}(F_I(\mathbf{x})) + V_{\mathcal{D}'}(F_I(\mathbf{x})) \quad (2.15)$$

[36, S. 376].

Aus den Formeln (2.11) und (2.15) folgt dann, dass

$$\mathbb{E}_{\mathcal{D}'}[F_I(\mathbf{x})] = \mathbb{E}_{\mathcal{D}}[F(\mathbf{x})] , \quad (2.16)$$

und somit gilt, dass

$$B_{\mathcal{D}'}(F_I(\mathbf{x})) = (\mathbb{E}_{\mathcal{D}}[F(\mathbf{x})] - \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}])^2 , = B_{\mathcal{D}}(F(\mathbf{x})) \quad (2.17)$$

also der Bias der gesamten Committee Machine  $B_{\mathcal{D}'}(F_I(\mathbf{x}))$  genauso groß ist, wie der Bias eines einzelnen Experten  $B_{\mathcal{D}}(F(\mathbf{x}))$ . Der Bias-Fehleranteil ändert sich durch das Ensemble Averaging also nicht. In ähnlicher Weise zeigt [36, S. 376f.], dass der

Varianz-Fehleranteil beim Ensemble Averaging maximal so groß ist wie bei einem einzelnen Experten:

Mit Gleichung (2.16) lässt sich die Varianz der Committee Machine als

$$V_{\mathcal{D}'}(F_I(\mathbf{x})) = \mathbb{E}_{\mathcal{D}'} \left[ F_I(\mathbf{x})^2 \right] - (\mathbb{E}_{\mathcal{D}} [F(\mathbf{x})])^2 \quad (2.18)$$

beschreiben [36, S. 377].

Und analog ist

$$V_{\mathcal{D}}(F_I(\mathbf{x})) = \mathbb{E}_{\mathcal{D}} \left[ F(\mathbf{x})^2 \right] - (\mathbb{E}_{\mathcal{D}} [F(\mathbf{x})])^2 \quad (2.19)$$

die Varianz eines Experten [36, S. 376].

Da das quadratische Mittel von  $F(x)$  über den gesamten Raum  $\mathcal{D}$  mindestens genauso groß sein muss wie das quadratische Mittel der Committee Machine Funktion  $F_I(x)$  über den Restraum  $\mathcal{D}'$ , also

$$\mathbb{E}_{\mathcal{D}} \left[ F(\mathbf{x})^2 \right] \geq \mathbb{E}_{\mathcal{D}'} \left[ F_I(\mathbf{x})^2 \right] \quad (2.20)$$

gilt, ist der Minuend in Gleichung (2.19) größer oder gleich dem Minuend in Formel (2.18). Da in beiden Gleichungen der selbe Subtrahend verwendet wird folgt, dass der Varianz-Fehleranteil der Committee Machine maximal so groß ist wie der Varianz-Fehleranteil eines einzelnen Experten. Demnach lässt sich die Ungleichung

$$V_{\mathcal{D}'}(F_I(\mathbf{x})) \leq V_{\mathcal{D}}(F_I(\mathbf{x})) \quad (2.21)$$

aufstellen [36, S. 377].

Die Schlussfolgerung ist, dass beim Ensemble Averaging der Bias gleich bleibt und die Varianz maximal so groß ist wie ohne Committee Machine. Der Gesamtfehler kann sich also nicht verschlechtern, was der Erkenntnis vom Beginn dieses Abschnitts mit den Gleichungen (2.9) und (2.10) entspricht. Da beim Overfitting der Gesamtfehler aus einem geringen Bias- und einem höheren Varianz-Anteil besteht [37, S. 93f.], die Varianz aber durch das Ensemble Averaging in der Committee Machine reduziert wird, ohne dass der Bias erhöht wird, sollte es vorteilhaft sein, mit den einzelnen Experten zu overfitten [36, S. 377].

## 2.5.5 Modell-Auswahl

Wenn für eine Problemstellung eine große Zahl an Modellen erzeugt wird, stellt sich die Frage, wie die Modelle ausgewählt werden sollen, die schließlich das Ensemble bilden. Denn der naive Ansatz, die  $N$  Modelle auszuwählen, die die besten Metriken erzeugen führt nicht zwingend zum besten Ensembling-Ergebnis. Wie bereits erläutert, ist dafür zusätzlich eine hinreichende Diversität der einzelnen Modelle nötig.

Caruana et al. [38] stellen verschiedene Strategien zur Modell-Auswahl aus einer sogenannten Modell-Bibliothek vor. Die Grundlegende Vorgehensweise sieht dabei wie folgt aus.

1. Erstelle ein leeres Ensemble.
2. Füge dem Ensemble das Modell hinzu, mit dem das Ensemble die beste Metrik auf den Validierungsdaten erreicht. Dieses Modell wird aus der Bibliothek entfernt.
3. Wiederhole Schritt 2 für eine feste Anzahl Iterationen oder bis alle Modelle genutzt wurden.

Dieses Vorgehen ist allerdings anfällig für eine Überanpassung an den Datensatz, und wenn keine Gegenmaßnahmen getroffen werden, kann es passieren, dass ab einem gewissen Punkt Modelle gezogen werden, die das Ensemble verschlechtern. Daher präsentieren die Autoren drei Erweiterungen, die diese Probleme lindern sollen.

Die erste Möglichkeit darin besteht, dass die Modelle in Schritt 2 mit Zurücklegen gezogen werden, anstatt aus der Bibliothek entfernt zu werden. Dies verhindert, dass dem Ensemble Modelle hinzugefügt werden, die der Gesamtperformanz schaden.

Darüber hinaus kann das Ensemble mit den  $N$  besten Modellen initialisiert werden. Da diese bereits gute Metriken aufweisen, wird es für die anschließende Suche schwieriger, Modelle zu finden, die ein Overfitting begünstigen.  $N$  könnte dabei etwa 5 – 25 Modelle umfassen.

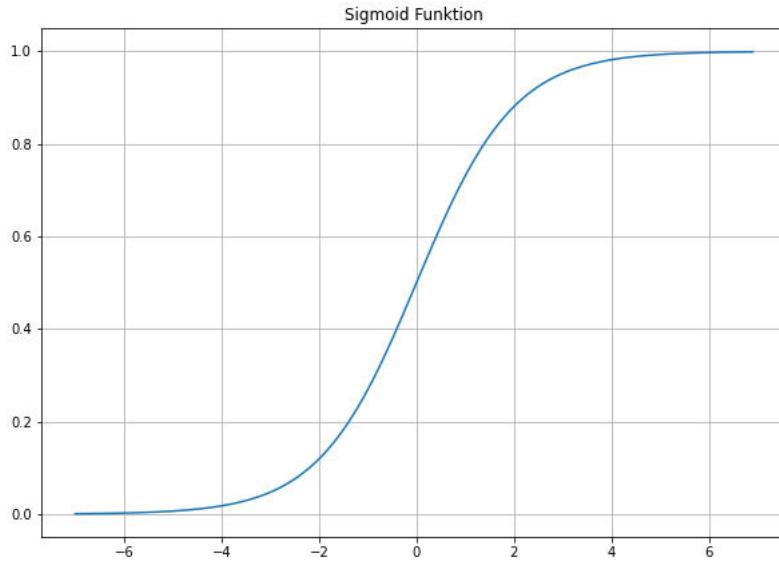
Die dritte Methode bezeichnen die Autoren als *Bagged Ensemble Selection*. Diese besteht darin, für Schritt 2 nicht die gesamte Bibliothek, sondern ein zufälliges Subset zu verwenden. Dadurch wird die Wahrscheinlichkeit gesenkt, dass Modell-Kombinationen gefunden werden können, die zu einer Überanpassung führen.

## 2.6 Wichtige Metriken

Es existieren mehrere Metriken um die Güte eines Klassifikators zu beurteilen. Dabei ist es wichtig, zur vorhandenen Problemstellung die passende Metrik zu wählen. Ansonsten kann es zu Trugschlüssen kommen und die Güte falsch eingeschätzt werden. Wie später noch dargelegt wird, ist die entscheidende Metrik im Bereich der Klassifikation von Thorax-Röntgenbildern die sogenannte *ROC-AUC*. Im Folgenden wird zu dieser Metrik hingeführt und dabei aufgezeigt, wo die Probleme bei der Verwendung von Alternativen liegen.

### 2.6.1 Vom Modelloutput zur Klasse

Zunächst muss festgehalten werden, wie aus dem Modelloutput die Vorhersage einer Klasse entsteht. Dafür werden die Modellausgaben als Klassenwahrscheinlichkeiten interpretiert. Betrachten wir zunächst eine Single-Label Multi-Class Klassifikation, bei der genau eines der möglichen Label positiv und der Rest negativ ist (vgl. Kapitel 3.1).



**Abbildung 2.11:** Plot der Sigmoid-Funktion für  $x \in \{-7, \dots, 7\}$ .

Damit dies möglich ist, müssen die Ausgaben zwei Eigenschaften erfüllen: Der erlaubte Wertebereich ist  $[0, 1]$  und da ein Objekt genau einer Klasse zugeordnet werden kann, soll die Summe der Wahrscheinlichkeiten 1 ergeben [39, S. 175f.]. Um dies sicherzustellen kann die sogenannte *softmax*-Funktion

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum e^x} \quad (2.22)$$

verwendet werden [39, S. 176]. Das Label, dem die höchste Wahrscheinlichkeit zugewiesen wird, kann dann als Vorhersage des Modells betrachtet werden [39, S. 179].

Das der Arbeit zugrundeliegende Problem ist allerdings eine Multi-Label Klassifikation. In diesem Fall ist die Anwendung der softmax-Funktion nicht sinnvoll, da mehrere statt nur einem einzigen Label positiv sein können. Damit fällt die Anforderung weg, dass die Summe der vorhergesagten Wahrscheinlichkeiten 1 ergeben muss. Somit ist es möglich, statt der softmax-Funktion die *sigmoid*-Funktion

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2.23)$$

zu benutzen, die ebenfalls zusichert, dass der Wertebereich  $[0, 1]$  ist [26, S. 134]. Diese wird in Abbildung 2.11 gezeigt.

Die Ausgaben der sigmoid-Funktion werden als Wahrscheinlichkeiten interpretiert und anhand eines Schwellwerts wird entschieden, ob das Label positiv ist. Ein intuitiver Schwellwert ist 0.5 bzw. 50%. Damit wäre die Label-Vorhersage  $\hat{y}_i$  für das  $i$ -te Label

		Tatsächliche Klasse $y$	
		1	0
Vorhergesagte Klasse $\hat{y}$	1	TP	FP
	0	FN	TN

Abbildung 2.12: Darstellung einer Konfusionsmatrix.

bei Modelloutput  $o_i$

$$\hat{y}_i = \begin{cases} 1 & \text{wenn } \text{sigmoid}(o_i) > 0.5, \\ 0 & \text{sonst.} \end{cases}$$

## 2.6.2 Konfusionsmatrix

Betrachtet man jedes mögliche Label als binäre Klassifikation lassen sich vier unterschiedliche Szenarien feststellen. Mit der Modellvorhersage  $\hat{y}$  und der tatsächlichen Klasse  $y$ :

- $y = 0$  und  $\hat{y} = 0$ : Das Modell hat die negative Klasse richtig klassifiziert, dies ist ein True Negative (TN). Oder
- $y = 0$  und  $\hat{y} = 1$ : Das Modell hat die negative Klasse falsch klassifiziert, was False Positive (FP) genannt wird. Oder
- $y = 1$  und  $\hat{y} = 1$ : Das Modell hat die positive Klasse richtig klassifiziert, es liegt ein True Positive (TP) vor. Und zuletzt
- $y = 1$  und  $\hat{y} = 0$ : Das Modell hat die positive Klasse falsch klassifiziert, was einem False Negative (FN) entspricht.

Die Kategorisierung zu TN, TP, FN und FP wird im Folgenden verwendet, um weitere Metriken zu definieren.

Die Anzahl oder der prozentuale Anteil der jeweiligen Szenarien wird häufig in einer Tabelle dargestellt, die als „Konfusionsmatrix“ oder „binäre Kontingenztafel“ [40] bekannt ist.

Überträgt man diese Darstellung auf eine Multi-Class Klassifikation und trägt als Zeilen und Spalten die jeweilige tatsächliche und vorhergesagte Klasse auf, lässt sich auch schnell sehen, welche Klassen dem Modell einfach oder schwierig fallen,

		Tatsächliche Klasse $y$	
		1	0
Vorhergesagte Klasse $\hat{y}$	1	0	0
	0	460	100.000

**Abbildung 2.13:** Konfusionsmatrix für den Klassifizierer der für die Klassifikation von COVID-19 pauschal die negative Klasse vorhersagt.

also gut bzw. schlecht klassifiziert wurden oder ob gar bei gewissen Labeln häufig Verwechslungen auftreten. Mit derartigen Erkenntnissen kann das Modell oder das Trainingsvorgehen angepasst werden um die Resultate zu optimieren.

### 2.6.3 Genauigkeit

Die Genauigkeit (engl. *accuracy*) ist eine der einfachsten Metriken und bezeichnet den Anteil der richtig klassifizierten Objekte. Sie lässt sich als

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.24)$$

berechnen [41]. Der Wertebereich der Genauigkeit ist  $[0, 1]$ , sie wird aber häufig prozentual angegeben.

Obwohl eine hohe Genauigkeit generell erstrebenswert ist, kann diese Metrik gerade im medizinischen Bereich schnell einen falschen Eindruck erwecken [41].

Nehmen wir aus Gründen der Aktualität an, wir möchten anhand von Thorax-Röntgenbildern COVID-19 erkennen. Da die Krankheit verhältnismäßig neu ist, ist zu erwarten, dass noch keine großen Datensätze dazu existieren. Nach bestem Wissen des Autors beinhaltet der größte öffentlich verfügbare COVID-19-Datensatz zum Zeitpunkt des Schreibens etwa 460 Thorax-Röntgenbilder von COVID-19-Patienten [42]. Zusätzlich verwenden wir für das Gedankenexperiment einen größeren Datensatz mit 100.000 Thorax-Röntgenbildern, die sowohl gesunde Lungen, als auch verschiedene Krankheiten (außer COVID-19) enthalten. Für die Klassifikation nach COVID-19 gibt es dann in den kombinierten Daten 460 positive und 100.000 negative Trainingsbeispiele. Gehen wir nun von einem Modell aus, dass pauschal (sogar ohne Betrachtung des Bildes) vorhersagt, dass der Patient kein COVID-19 hat. Damit ergibt sich folgende Konfusionsmatrix:

Damit würde sich die Accuracy analog zu Gleichung (2.24) für das Beispiel als

$$\text{accuracy} = \frac{0 + 100.000}{0 + 100.000 + 0 + 460} = 99.5\%$$

berechnen. Es ist also nicht einmal nötig, die Bilder zu betrachten um eine Accuracy von über 99% zu erreichen. Dieses Extrembeispiel zeigt, dass ein Datensatz mit starker Imbalance diese Metrik ad absurdum führt.

Des Weiteren lässt sich argumentieren, dass gerade im medizinischen Umfeld False Negatives viel schwerwiegender sein können als False Positives, oder je nach Situation auch das Gegenteil der Fall ist. Würde das Modell dem Patienten fälschlicherweise sagen, dass er kein COVID-19 hat, wäre das Risiko deutlich höher, dass er den Virus weiter verbreitet und auch sich selbst in Gefahr bringt. Hier könnten zu viele False Negatives schwerwiegende Folgen haben. Hingegen würde ein False Positive, also wenn das Modell einem eigentlich negativen Patienten sagt, er hätte COVID-19, zunächst einmal zu übervorsichtigen Schutzmaßnahmen wie Isolation führen, aber zumindest kein unmittelbar größeres Gesundheitsrisiko verursachen.

Auf der anderen Seite könnten False Positives schwerwiegender sein, wenn sie zum Beispiel zu riskanten medizinischen Maßnahmen führen [41]. Dies veranschaulicht, dass weitere Metriken notwendig sind, um die Güte eines Modells passend zu beurteilen.

## 2.6.4 True Positive Rate

Die True Positive Rate (TPR) legt den Fokus darauf, möglichst viele positive Objekte richtig zu klassifizieren [40]. Sie lässt sich berechnen als

$$TPR = \frac{TP}{TP + FN} \tag{2.25}$$

und wird auch als Recall oder Sensitivity bezeichnet [40]. Ihr Wertebereich ist  $[0, 1]$ . Gehen wir zurück zu unserem Extrembeispiel. Die TPR berechnet sich hierfür als

$$TPR = \frac{0}{0 + 460} = 0.$$

Während die Accuracy also nahezu perfekt ist (99,5%), offenbart die TPR von 0, dass der Klassifizierer auf keinen Fall eingesetzt werden sollte. Daher ist diese Metrik besonders im medizinischen Umfeld von großer Bedeutung [40].

## 2.6.5 False Positive Rate

Die False Positive Rate (FPR) (auch als Fallout oder False Alarm bezeichnet) misst den Anteil der False Positives, also wie viele tatsächlich negative Objekte als positiv klassifiziert wurden und wird über

$$FPR = \frac{FP}{FP + TN} \tag{2.26}$$

definiert [40]. Wie bei der TPR hat auch die FPR den Wertebereich  $[0, 1]$ . Da sie einen Fehleranteil bemisst, sind allerdings niedrigere Werte besser. Für unser Beispiel wäre die  $FPR = \frac{0}{0+100.000} = 0$ . Auch diese Metrik versteckt also ähnlich zur Accuracy die Ineffektivität des Klassifizierers.

## 2.6.6 Spezifität

Die Spezifität (engl. *specificity*)

$$\text{specificity} = \frac{TN}{TN + FP} \quad (2.27)$$

ist das Pendant der TPR bzw. Sensitivity für negative Klassen [41].

Aus den Gleichungen (2.26) und (2.27) wird schnell klar, dass gilt

$$FPR + \text{specificity} = \frac{FP}{FP + TN} + \frac{TN}{TN + FP} = 1. \quad (2.28)$$

Daher wird die FPR auch manchmal über  $FPR = 1 - \text{specificity}$  definiert.

Die Spezifität für unser Extrembeispiel, bei dem nur negative Klassen vorhergesagt werden, berechnet sich als

$$\text{specificity} = \frac{100.000}{100.000 + 0} = 1.$$

Da die Spezifität ein Maß für die Erkennung von negativen Objekten ist, ist es offensichtlich, dass ein Klassifizierer, der immer die negative Klasse vorhersagt, eine perfekte Spezifität erreicht.

Diese Beispiele zeigen, dass es äußerst wichtig ist, die richtige Metrik zur Gütebewertung einzusetzen. Außerdem können gewisse Metriken ein verzerrtes Bild liefern, wenn sie alleine betrachtet werden. Das verwendete Extrembeispiel zeigt, dass es sinnvoll sein kann, sowohl die TPR als auch die FPR gleichzeitig zu berücksichtigen.

## 2.6.7 Receiver Operating Characteristic (ROC)

In Abschnitt 2.6.1 wurde dargestellt, dass der Output des Klassifizierers mittels eines Schwellwerts auf die Klassen abgebildet wird. Der dort gewählte Schwellwert von 0.5 ist zwar zunächst intuitiv, aber dennoch willkürlich. Ein höherer oder niedriger Schwellwert könnte zu besseren Resultaten führen. Genauso könnte man aufgrund der Auswirkungen der entweder positiven oder negativen Klasse voraussetzen, dass sich der Klassifizierer sehr sicher ist und daher einen höheren respektive niedrigeren Schwellwert einsetzen.

Je höher der Schwellwert für die positive Klasse ist, desto höher würden tendenziell die False Negatives, da mehr Objekte als negativ klassifiziert werden. Dies gilt analog für niedrige Schwellwerte und False Positives.

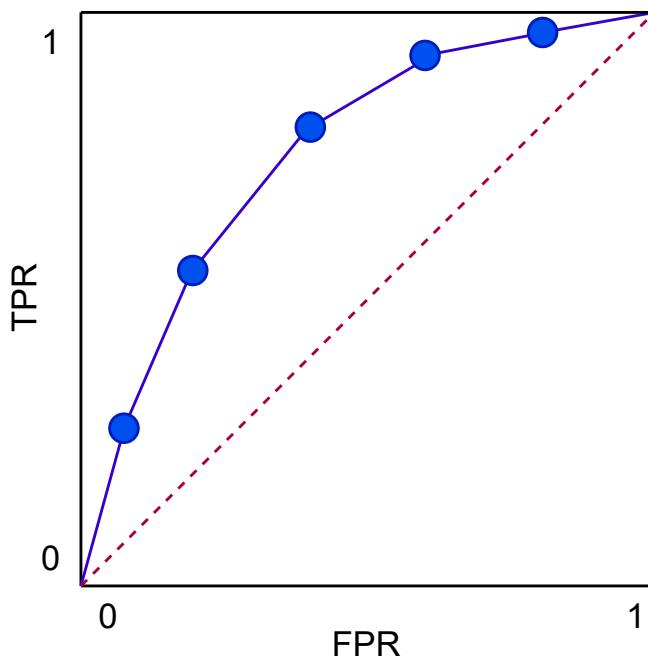


Abbildung 2.14: Plot einer fiktiven ROC-Kurve.

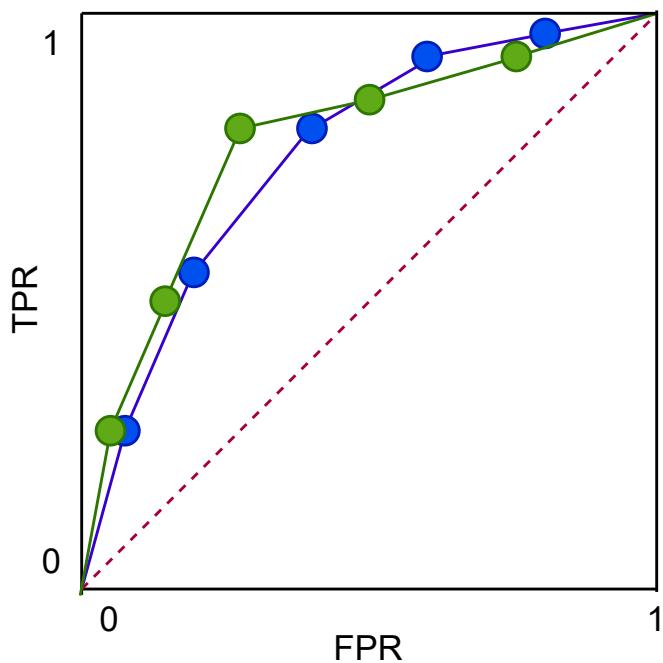
Da sich mit dem Schwellwert im Allgemeinen auch die zugewiesenen Klassen ändern, sind auch die TPR und die FPR von Änderungen betroffen. Dies kann es schwierig machen, Klassifizierer zu vergleichen und motiviert die Receiver Operating Characteristic (ROC) Kurve [41]. Für diese Metrik werden die Wertepaare (FPR, TPR) zu den jeweiligen Schwellwerten auf der ( $x, y$ )-Achse aufgetragen. Somit ist die ROC Kurve unabhängig vom gewählten Schwellwert. Zudem ist sie unabhängig von der Häufigkeit der Klassen, da sie sowohl die TPR als auch die FPR berücksichtigt. In Abbildung 2.14 wird der Plot einer solchen ROC Kurve gezeigt.

Je niedriger die FPR und je höher die TPR, desto besser ist ein Klassifizierer. Für den Kurvenverlauf bedeutet dies, dass ein Klassifizierer besser ist, je weiter links oben dessen ROC Kurve verläuft [40]. Ein naiver Klassifizierer, der mit gleicher Wahrscheinlichkeit zufällige Vorhersagen erzeugt, hätte unabhängig vom Schwellwert die gleiche TPR wie FPR. Somit lässt sich die Winkelhalbierende des Koordinatensystems als schlechtestmögliche ROC-Kurve verstehen [40]. Des Weiteren verläuft jede ROC-Kurve durch die Punkte  $(0,0)$  und  $(1,1)$ , da diese jeweils den Extremfällen entsprechen, alle Objekte als negativ ( $TPR = FPR = 0$ ) respektive als positiv ( $TPR = FPR = 1$ ) zu klassifizieren [43].

### 2.6.8 Area Under the Curve (AUC)

Wenn man versucht mehrere Klassifizierer anhand ihrer ROC Kurve zu vergleichen kann es zu Schwierigkeiten kommen, solange nicht eine Kurve klar bei jedem Schwellwert über der anderen liegt. Ein derartiges Beispiel findet sich in Abbildung 2.15.

Die Area Under the Curve (AUC) schafft auf einfache Weise Abhilfe. Wie der Name bereits verrät wird dabei die Fläche unter einer – zunächst beliebigen – Kurve berechnet.



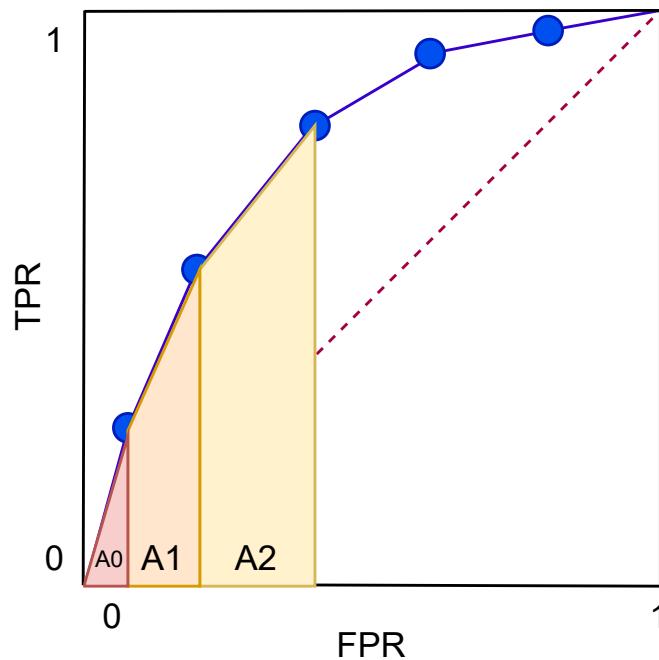
**Abbildung 2.15:** Plot zweier fiktiver ROC-Kurven, die von unterschiedlichen Klassifizierern erzeugt werden. Es ist nicht trivial ersichtlich, ob der grüne oder der blaue Klassifizierer besser ist.

Meistens bezieht sich dies aber auf die ROC Kurve, weswegen ROC-AUC, AUROC oder AUC-ROC die genaueren Bezeichnungen für diese Metrik ist. Da typischerweise in den Veröffentlichungen zu Deep Learning für Thorax-Röntgenbilder allerdings nur von AUC gesprochen wird, wird diese Nomenklatur für die vorliegende Arbeit übernommen.

Statt die einzelnen ROC-Kurven zu vergleichen, vergleicht man also die Fläche unter ihnen [43]. Je größer diese Fläche und somit der AUC Wert, desto besser ist der Klassifizierer. Da die TPR und FPR einen Wertebereich von  $[0, 1]$  haben, spannen sie ein Quadrat mit der Fläche 1 auf, woraus sich schließen lässt, dass auch der AUC einen Wertebereich von  $[0, 1]$  hat [43]. Wie oben erwähnt entspricht die ROC-Kurve einer zufälligen Klassifizierung der Winkelhalbierenden, was einem AUC-Wert von 0.5 entsprechen würde [43]. Demnach sollte kein Klassifizierer einen AUC-Wert unter 0.5 aufweisen, da er ansonsten schlechter als der Zufall wäre.

Die AUC kann auch aus Sicht des *Ranking* betrachtet werden kann. In dieser Interpretation entspricht die AUC der Wahrscheinlichkeit, dass ein zufällig gezogenes positives Beispiel einen höheren Rang (oder allgemein *Score*) erhält, als ein zufälliges negatives Beispiel [44]. Hat ein Modell, das Lungenentzündungen erkennen soll also eine AUC von 0.9, ist die Wahrscheinlichkeit 90%, dass die Modellvorhersage für ein zufälliges positives Beispiel *näher* an 1 liegt als für ein zufälliges negatives Beispiel. Dies veranschaulicht, warum der AUC-Wert im medizinischen Bereich eine wichtige Metrik darstellt.

Die Berechnung der AUC erfolgt durch iteratives Aufsummieren der Flächen der Trapeze, die zwischen zwei aufeinanderfolgenden Wertepaaren ( $\text{FPR}_i, \text{TPR}_i$ ) und



**Abbildung 2.16:** Visualisierung des Berechnungsprozesses für den ROC-AUC Wert. Dabei wird iterativ die Formel zur Berechnung der Trapezfläche aus Gleichung (2.29) auf die Fläche, die sich aus FPR und TPR zwischen zwei Schwellwerten ergibt, angewendet.

$(FPR_{i+1}, TPR_{i+1})$  entstehen, und anschließendem Dividieren durch die mögliche Fläche um auf den Wertebereich  $[0, 1]$  zu skalieren [43]. Dabei stellt  $FPR_i$  den Wert der False Positive Rate für den  $i$ -ten Threshold dar, analog für  $TPR_i$ .

Sei  $(x_1, y_1)$  das Wertepaar  $(FPR_i, TPR_i)$  und  $(x_2, y_2)$  das Wertepaar  $(FPR_{i+1}, TPR_{i+1})$ , dann lässt sich die Fläche des Trapezes zwischen den ROC-Werten  $i$  und  $i + 1$  über

$$A_{\text{trapez}} = |x_1 - x_2| \times \left( \frac{1}{2}(y_1 + y_2) \right) \quad (2.29)$$

berechnen [43]. Dieses Vorgehen wird in Abbildung 2.16 visualisiert.

Mit Hinblick auf das Ziel der vorliegenden Arbeit und dem Einsatz von Ensembling-Methoden, sei darauf hingewiesen, dass die Berechnung der AUC nicht sinnvoll ist, wenn ein Hard Voting eingesetzt wurde. Denn dieses führt dazu, dass die Vorhersagen nicht mehr als Wahrscheinlichkeit interpretiert werden können.

## 2.6.9 Erweiterung auf Multi-Class Probleme

Die bisherigen Definitionen für die Metriken beruhen allesamt zunächst auf einem binären Klassifikationsproblem. Sie lassen sich jedoch durch Mittelwertbildung leicht auf Multi-Class Klassifikationen anwenden. Dabei unterscheidet man zwei Herangehensweisen: Das Micro-Averaging und das Macro-Averaging [45].

Für das Labelset  $\mathcal{L}$  wird das Micro-Averaging einer Metrik  $M(tp, tn, fp, fn)$  mit den True Positives  $tp_\lambda$  für die Klasse  $\lambda$  und analog für TN, FP und FN über

$$M_{micro} = M \left( \sum_{\lambda=1}^{|\mathcal{L}|} tp_{\lambda}, \sum_{\lambda=1}^{|\mathcal{L}|} tn_{\lambda}, \sum_{\lambda=1}^{|\mathcal{L}|} fp_{\lambda}, \sum_{\lambda=1}^{|\mathcal{L}|} fn_{\lambda} \right) \quad (2.30)$$

berechnet [45]. Es werden also die TP, TN, FP und FN aller Klassen gleichzeitig betrachtet und die Metrik nur einmal berechnet.

Dahingegen wird beim Macro-Averaging

$$M_{macro} = \frac{1}{|\mathcal{L}|} \sum_{\lambda=1}^{|\mathcal{L}|} M(tp_{\lambda}, tn_{\lambda}, fp_{\lambda}, fn_{\lambda}) \quad (2.31)$$

die Metrik für jede Klasse einzeln berechnet und über diese Ergebnisse der Mittelwert gebildet [45].

# Kapitel 3

## Multi-Label Klassifizierung

Im Folgenden soll die Multi-Label Klassifizierung, die bei der Thorax-Röntgenbilder Klassifikation vorherrschend ist, vorgestellt werden.

Generell gesprochen geht es bei einer Klassifizierung darum, ein Objekt  $\mathbf{x} = (x_1, \dots, x_a)$  mit  $a$  Feature aus dem Input-Raum  $\mathcal{X}$  mittels eines Klassifizierers  $f : \mathcal{X} \rightarrow \mathcal{Y}$  auf den Output-Raum  $\mathcal{Y}$  abzubilden [46, S. 11f.].

Die Menge aller möglichen Klassen wird von Mencia [46, S. 14] bezeichnet als  $\mathcal{L} := \{\lambda_0, \dots, \lambda_n\}$  mit  $n = |\mathcal{L}|$ . Darüber hinaus bezeichnen die Autoren mit der Menge  $P$  alle positiven Klassen eines Objekts, also alle Label, die dem Objekt korrekterweise zugeordnet werden können, sowie mit  $N$  die Menge aller negativen Klassen. Der Output-Raum  $\mathcal{Y}$  ist dann ein  $n$ -dimensionaler Vektor, der angibt ob die Klasse vorliegt oder nicht [46, S. 14]

$$\mathcal{Y} := \{0, 1\}^n , \quad y = (y_1, \dots, y_n) \in \mathcal{Y} , \quad y_i := \begin{cases} 1 & \text{wenn } \lambda_i \in P \text{ positiv ,} \\ 0 & \text{sonst .} \end{cases}$$

Daraus lassen sich nach [46, S. 14] festhalten, dass die Menge aller positiven Klassen  $P$  eine Teilmenge aller Klassen  $\mathcal{L}$  ist,  $P \subseteq \mathcal{L}$ . Daraus ergibt sich, dass die Menge aller negativen Klassen  $N$   $\mathcal{L}$  ohne  $P$  ist,  $N := \mathcal{L} \setminus P$ .  $P$  und  $N$  ergeben somit gemeinsam  $\mathcal{L}$ ,  $P \cup N = \mathcal{L}$  und haben keine Gemeinsamen Elemente,  $P \cap N = \emptyset$ .

### 3.1 Klassifizierungsarten

Klassifizierungsprobleme lassen sich nach zwei Eigenschaften deklinieren. Einerseits anhand der *Kardinalität*, also der Anzahl möglicher positiver Klassen  $|P|$ , und andererseits durch die *Dimension*, also die Anzahl der Klassen  $|\mathcal{L}|$  [46, S. 15]. Basierend darauf lässt sich unter anderem zwischen Binärklassifikation, Multi-Class Klassifikation und Multi-Label Klassifikation unterscheiden. Neben dem Begriff „Klasse“ wird auch der des „Label“ verwendet. Zwischen diesen existiert jedoch keine klare Abgrenzung, wenngleich Label häufiger im Kontext der Multi-Label Klassifikation genutzt wird.

### 3.1.1 Binärklassifikation

Bei der Binärklassifikation ist der Output-Raum ein einzelnes Element. Es geht darum zu entscheiden, ob eine einzige Klasse vorhanden ist oder nicht. Formal wird dabei zunächst also nicht zwischen einer positiven und einer negativen Klasse unterschieden, vielmehr entspricht die negative Klasse der Absenz des positiven Labels [46, S. 16]. Das heißt, dass die Menge der positiven Klassen genau ein Element enthält  $P = \lambda_1$  und die leere Menge  $P = \emptyset$  als negative Klasse interpretiert wird. Damit ist die Binärklassifikation zwingend eine single-label Klassifikation. Demnach lässt sich die Binärklassifikation nach [46, S. 16] definieren über

$$|\mathcal{L}| = 1, \quad \mathcal{Y}_{\text{binary}} := \mathcal{Y}^1.$$

Typischerweise wird bei Machine-Learning Projekten ein binäres Label mit einem Bit kodiert. Demnach entspricht die negative Klasse  $y = 0$  und die positive Klasse  $y = 1$ . In selteneren Fällen wird dies auch als  $y = -1$  und  $y = 1$  abgebildet. Streng genommen liegt hier laut [46, S. 16] demnach keine Binärklassifikation vor, sondern eine Multi-Class Klassifikation, da sich die Binärklassifikation nur mit dem Vorhandensein der positiven Klasse beschäftigt. Die negative Klasse ergibt sich jedoch implizit aus der Abwesenheit der positiven Klasse. Daher wird dieses Szenario üblicherweise als Binärklassifikation oder Zwei-Klassen-Problem bezeichnet. Bei der Verwendung Neuronaler Netze ist es zudem zwingend notwendig, auch der negativen Klasse einen Zahlenwert zuzuweisen, um überhaupt ein Training mit Backpropagation durchführen zu können.

Die Binärklassifikation beantwortet zusammengefasst die Frage „Trifft diese eine Klasse auf das Objekt zu?“

### 3.1.2 Multi-Class Klassifikation

Ist die Dimension  $|\mathcal{L}| = n > 1$  spricht man von einer *Multi-Class Klassifikation* [46, S. 17]. Meistens handelt es sich gleichzeitig um eine Single-Label Klassifikation. Das heißt, dass genau ein  $y_i = 1$ , also exakt ein Label, positiv ist. Somit lässt sich über

$$\mathcal{Y}_{\text{Multi-Class}}^n \subseteq \mathcal{Y}^n, \quad \forall \mathbf{x}. |P_x| = 1, \quad \mathcal{Y}_{\text{Multi-Class}}^n = \{y \in \mathcal{Y}^n \mid |y| = 1\}$$

die Single Label Multi-Class Klassifikation definieren [46, S. 17].

Anders formuliert geht es dabei um die Fragestellung: „Welche einzige der  $n$  Klassen trifft auf das Objekt zu?“

### 3.1.3 Multi-Label Klassifikation

Die *Multi-Label Klassifikation* ist eine Sonderform der Multi-Class Klassifikation. Genau genommen könnte man sie deshalb auch als Multi-Label Multi-Class Klassifikation bezeichnen [46, S. 17]. Das besondere ist, dass hierbei nicht genau ein Label, sondern eine beliebige Anzahl der möglichen Label  $|\mathcal{L}| = n > 1$  positiv sein kann. Demnach lässt sich eine solche Multi-Label Klassifikation über die Eigenschaften

$$\mathcal{Y}_{\text{Multi-Label}}^n := \mathcal{Y}^n, \quad \forall \mathbf{x}. 0 \leq |P_x| \leq n, \quad \mathcal{Y}_{\text{Multi-Class}}^n = \{y \in \mathcal{Y}^n \mid |y| = 1\}$$

definieren [46, S. 17].

Die Multi-Label Klassifikation beschäftigt sich also mit der Frage „Welche beliebige Anzahl der  $n$  Klassen trifft auf das Objekt zu?“ Die durchschnittliche Anzahl der positiven Label pro Datenpunkt wird als *Kardinalität* bezeichnet [47].

### 3.1.4 Hierarchische Klassifikation

Für das konkrete Klassifizierungsproblem der vorliegenden Arbeit lässt sich zwischen einigen Labeln, also Krankheiten bzw. Krankheitsindikatoren, eine hierarchische Beziehung feststellen (siehe auch Abschnitt 7.3). In einem solchen Fall kann man das Problem auch als *Hierarchische Klassifikation* betrachten und die Label als gerichteten Graph bzw. Baum darstellen [46, S. 22]. In einem hierarchischen Multi-Label Problem sind im Allgemeinen alle Label positiv, die zwischen einem Eltern und einem positiven Blatt-Knoten liegen [46, S. 22].

Diese Informationen über die Struktur der Label werden häufig aber nicht gesondert verwendet, da dies zusätzliche Maßnahmen erfordern würde [46, S. 22f.]. Eine Berücksichtigung der Hierarchie könnte zwar Vorteile gegenüber der Betrachtung als herkömmliche Multi-Label Klassifikation haben, es lässt sich aber auch argumentieren, dass ein starker Klassifizierer diese Beziehungen selbstständig erlernen kann oder sollte [46, S. 22f.].

Nichtsdestotrotz wird sich ein Teil dieser Arbeit (Kapitel 7.3) auch mit der Hierarchie der Pathologien beschäftigen und versuchen, diese als Domänenwissen zu nutzen, um die Klassifikation zu verbessern .

## 3.2 Problemtransformationen

Viele typische Machine-Learning-Methoden sind in ihrer ursprünglichen Form nur für Multi-Class Klassifikationen geeignet. Um mit ihnen dennoch eine Multi-Label Klassifikation durchführen zu können gibt es verschiedene Ansätze, die sich grob einteilen lassen in das Anpassen eines Klassifizierers oder das Transformieren der Problemstellung von einer Multi-Label zu mehreren Multi-Class Klassifizierungen [46, S. 36]. Das Transformieren ermöglicht die Nutzung eines beliebigen Klassifizierers, während das Anpassen für jeden Klassifizierer spezifisch vorgenommen werden muss. Neuronale Netze sind ohne besondere Anpassungen zur Multi-Label Klassifikation imstande. Da in Abschnitt 7.5 allerdings auch eine Logistische Regression zur Klassifizierung verwendet wird, werden im Folgenden die Möglichkeiten zur Problemtransformation prägnant vorgestellt.

### 3.2.1 Binary Relevance Decomposition

Die *Binary Relevance (BR) Decomposition* ist die häufigste Methode zur Problemtransformation [46, S. 53]. Dabei wird ein Multi-Label Problem mit  $n$  Klassen auf  $n$  Klassifizierer  $h_i$  aufgeteilt, die mit  $n$  binären Trainingsdatensätzen  $\mathcal{T}_{train}$  gleicher Länge  $m$  trainiert werden [46, S. 54]. Die  $\mathcal{T}_{train}$  haben dabei jeweils die gleiche Länge wie der originale Trainingsdatensatz, indem für jede Klasse  $i$  mit  $i = 1 \dots n$  aus den ursprünglichen

Trainingsdaten  $(x_j, y_j)$  für  $j = 1 \dots m$  Beispiele der Form  $(x_j, y_{j,i})$  erzeugt werden. Damit wird  $\mathcal{T}_{train}$  über

$$\mathcal{T}_{train}^i = \langle (x_1, y_{1,i}), \dots, (x_m, y_{m,i}) \rangle$$

definiert [46, S. 54]. Anders ausgedrückt wird für jede im Datensatz vorhandene Klasse ein gesonderter Klassifizierer trainiert.

Da bei den nun binären Trainingsdaten die Klassifikation zwischen dem Label  $i$  und den anderen Labeln  $l$  für  $l = 1 \dots n, l \neq i$  besteht, ist die BR auch als *one-against-all*, *one-against-rest* beziehungsweise *one-versus-all* und *one-versus-rest* bekannt [46, S. 54].

Jeder der  $n$  Klassifizierer  $h_i$  sagt dann für einen Datenpunkt  $\mathbf{x}$  vorher, ob er zur Klasse  $i$  gehört oder nicht. Die Multi-Label Vorhersage  $\hat{\mathbf{y}}$  setzt sich also zusammen als

$$\hat{\mathbf{y}} := (h_1(\mathbf{x}), \dots, h_n(\mathbf{x})) ,$$

was  $n$  einzelnen Vorhersagen entspricht [46, S. 54].

Da alle  $n$  Trainingsdatensätze die gleiche Länge haben wie der Originaldatensatz, ist die Berechnungskomplexität des Trainings  $n$  mal so groß. Genauso ist auch die Komplexität der Inferenz  $n$  mal so groß, da  $n$  Klassifizierer ausgewertet müssen [46, S. 57].

### 3.2.2 Label Powerset Transformation

Bei der *Label Powerset (LP) Transformation* wird das Multi-Label Problem in ein Multi-Class Problem überführt, indem jede in den Daten auftretende Label-Kombination  $P_j$  als eine Klasse betrachtet wird [46, S. 57]. Es muss demnach das neue Label-Set  $\mathcal{L}_{LP}$ ,

$$\mathcal{L}_{LP} := \{P_j \mid 1 \leq j \leq m\} ,$$

betrachtet werden, dessen Länge  $n' = |\mathcal{L}_{LP}|$  ist [46, S. 57].

Da der Klassifizierer somit immer alle zutreffenden Label als Kombination vorhersagt, kann die Vorhersage direkt verwendet werden. Daraus folgt aber auch, dass es dem LP-Klassifizierer nicht möglich ist, Label-Kombinationen vorherzusagen, die nicht in den Trainingsdaten vorkommen [46, S. 57]. Da für gewöhnlich die Test- und Trainingsdaten aber ähnlich genug sein sollten, stellt dies in der Praxis normalerweise kein größeres Problem dar [46, S. 58].

Zudem sollte berücksichtigt werden, dass die Anzahl der neuen Klassen  $n'$  im ungünstigsten Fall entweder der Anzahl der Trainingsdaten  $m$  oder der möglichen Kombinationen der Ausgangsklassen  $n$  entspricht:  $n'_{\text{worst-case}} = \min(m, 2^n)$  [46, S. 58].

### 3.2.3 Classifier Chain

Während die LP die Beziehung zwischen mehreren Labeln direkt berücksichtigt, wird dies bei der BR ignoriert, was dazu führen kann, dass ein BR-Klassifizierer auch unmögliche Label-Kombinationen vorhersagt [48]. Bei dem vorliegenden Anwendungsfall wäre ein markantes Beispiel, dass der Klassifizierer sowohl „No Finding“ als auch gleichzeitig widersprüchlich eine Krankheit, z.B. Pneumonie vorhersagt.

Die *Classifier Chain* (CC) kann als eine Erweiterung der BR bezeichnet werden, die versucht, sowohl die geringe Berechnungskomplexität der BR mit der Nutzung von Label-Beziehungen zu kombinieren [48].

Für eine CC wird wie bei der BR für jede Klasse ein eigener Klassifizierer trainiert, insgesamt also erneut  $n$  Stück. Jedoch werden für jeden Klassifizierer zu den Features zusätzlich die vorherigen Label hinzugefügt, wodurch er die Label-Beziehungen erlernen kann [48]. Der Klassifizierer  $h_j$ , der das Label  $j$  vorhersagen soll, erhält also neben den Features zusätzliche alle vorherigen Label  $l_1, \dots, l_{j-1}$  als Eingabe.

Die Vorhersage von  $h_j$  für den Datenpunkt  $x_i$  ist dann also die Wahrscheinlichkeit für  $l_j$  unter der Bedingung von  $x_i$  und den vorherigen Labels:  $Pr(l_j|x_i, l_1, \dots, l_{j-1})$  [48]. Dies erklärt die Bezeichnung als Kette (engl. *chain*), da die einzelnen Label verkettet werden. Die Gesamtvorhersage der CC ist somit die Vorhersage des letzten Klassifizierers.

Die Reihenfolge der Label hat bei diesem Ansatz offensichtlich eine relevante Bedeutung. Daher kann es sich anbieten, ein Ensemble aus CCs zu bilden, bei dem die Labelreihenfolge für das Training der einzelnen CC zufällig festgelegt wird und die Trainingsdaten je eine Teilmenge der Originaldaten sind [48]. Um die finale Vorhersage zu bilden, können die typischen Vorgehen des Ensemblings verwendet werden, wie sie in Kapitel 2.5 vorgestellt werden, beispielsweise ein Voting-Verfahren.

In den Experimenten von Read et al. [48] erzielte die CC bessere Resultate als die Alternativen und diese konnten durch Ensembling weiter verbessert werden.

### 3.2.4 RAkEL

Um die Probleme der LP-Methode zu entschärfen, die dadurch entstehen, dass die Anzahl aller möglichen Label-Kombinationen sehr groß sein können, wird von Tsoumacas et al. [45] mit der *RAndom k-labELsets* (RAkEL) eine neue Vorgehensweise präsentiert. Dabei werden nicht alle möglichen Label-Kombinationen berücksichtigt, sondern ein Ensemble gebildet, indem zufällige, kleinere Subsets mit  $k$  Labels erzeugt werden. Mit dem Ursprungslabelset  $L$  ergibt sich somit immer ein Subset  $Y$  mit  $Y \subseteq L$ , für das  $k = |Y|$  gilt. Die Größe aller Subsets  $L^k$  ergibt sich somit durch den Binomialkoeffizienten als

$$|L^k| = \binom{|L|}{k}.$$

Auf den Subsets wird dann ein LP-Klassifizierer trainiert, dessen Performanz durch ein kleineres Labelset positiv beeinflusst wird. Das Labelset, das für einen der  $m$  LP-Klassifizierer des Ensembles zufällig gezogen wurde, wird dabei nicht zurückgelegt [45]. Tsoumacas et al. [45] erzeugen die Ensemble-Vorhersage, indem sie die binären Vorhersagen mitteln und dies anschließend mittels eines Entscheidungsschwellwerts auf die positive bzw. negative Klasse abbilden. In ihren Experimenten mit mehreren Benchmark-Datensätzen performte die RAkEL-Methode besser als BR und LP.

## Kapitel 4

# Transfer Learning

In Abschnitt 2.4.1 wurden bereits die grundlegenden Bausteine für das Erstellen eines CNNs vorgestellt. Alleine durch die Variation einiger Hyperparameter wie der Schichtenanzahl, den verwendeten Kernelgrößen oder auch der Anordnung der Schichten lässt sich bereits eine Vielzahl unterschiedlicher CNN-Architekturen realisieren. Zusätzlich könnte man beispielsweise neben Convolution und Pooling weitere Schichttypen einführen oder auf andere Weise neue Architekturen erzeugen. Zwar kann man davon ausgehen, dass komplexere Architekturen aufgrund höherer Parameteranzahl zu leistungsfähigeren Netzen führen, allerdings liefert das naive Hinzufügen von Schichten offensichtlich nicht zwingend die beste Architektur und der Erfolg von neuen Schichten kann nicht sicher vorhergesagt werden. Stattdessen ist es nötig, Architekturvorschläge gegeneinander zu vergleichen.

Die ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [21] dient als Benchmark im Bereich der Bildverarbeitung.

Aus dieser gingen mehrere bekannte Architekturen hervor, die sich etablierten und zu einem de-facto Standard geworden sind. Dazu zählen beispielsweise AlexNet, VGG und ResNet [49]. Neben ResNets wird auch eine ähnliche Architektur, die DenseNets, häufig im Bereich der Thorax-Röntgenbilder-Klassifikation verwendet.

Wenn man ein Netzwerk auf einem sehr großen Datensatz trainiert hat, beispielsweise für die ILSVRC, könnte man versuchen, den dabei geleisteten Trainingsaufwand auf andere Probleme zu übertragen, was zum Transfer Learning führt.

Dieses Kapitel gibt zunächst Informationen zur ILSVRC, stellt dann die Architekturen ResNet und DenseNet vor, formuliert anschließend das Prinzip des Transfer Learnings und schließt mit einem Abschnitt über die Anwendung von Transfer-Learning für medizinische Bilddaten.

## 4.1 ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

Die ILSVRC [21] ist eine jährlich stattfindende Challenge, die zwischen 2010 und 2017 durchgeführt wurde und auf dem ImageNet [50] Datensatz beruht.

ImageNet orientiert sich an der hierarchischen, semantischen Struktur des WordNet [51] und ist daher in 12 Unterbäume aufgebaut, die insgesamt 3.2 Millionen Bilder enthalten [50]. Zu den Themen der Unterbäume gehören Tiere, Fahrzeuge, Möbel, Musikinstrumente, geologische Formationen, Werkzeuge, Blumen und Früchte, die jeweils hierarchisch in weitere Unterkategorien dekliniert werden [50]. Die Bilder sind diversifiziert, enthalten also verschiedene Varianten, Positionen, Orientierungen, etc. der Objekte und die Label können als sehr genau bzw. rein bezeichnet werden ( $\approx 99\%$  Precision) [50].

Die ILSVRC beinhaltete über die Jahre drei Aufgaben [21]. Für die *Image Klassifikation* soll eine Liste von im Bild vorhandenen Objekt-Kategorien vorhergesagt werden. Die *Einzelobjekt Lokalisation* erfordert eine Liste von im Bild vorhandenen Objekt-Kategorien und *eine* ihrer Positionen. Und die *Objekt-Detektion* besteht in der Vorhersage einer Liste von im Bild vorhandenen Objekt-Kategorien und *aller* ihrer Positionen.

Für die Challenge wurden 1.000 Kategorien aus dem ImageNet so ausgewählt, dass es keine Überlappungen gibt, also keine in der Hierarchie aufeinanderfolgenden Teilbäume genutzt werden [21]. Allerdings waren die Kategorien nicht jedes Jahr gleich. Während die Trainingsbilder direkt aus dem ImageNet genommen wurden, wurden die Validierungs- und Testdaten eigens für die ILSVRC aus dem Internet zusammengetragen, wofür das gleiche Vorgehen wie für das Erstellen der ImageNet Datenbank angewendet wurde [21]. Auch für das Labeling wurde genauso wie beim ImageNet vorgegangen, indem die Bilder über Amazon Mechanical Turk via Crowd-Sourcing gegen monetäre Anreize gelabelt wurden und das Label über einen Mehrheitsentscheid bestimmt wurde [21]. Insgesamt umfassten die Trainingsdaten etwa 1.2 Millionen, die Validierungsdaten 50.000 und die Testdaten 100.000 Bilder [21].

## 4.2 Transfer Learning für die Bildverarbeitung

Beim Transfer Learning soll ein Modell die Erkenntnis aus dem Training auf einem Datensatz  $D_1$  der Verteilung  $P_1$  auf einen Datensatz  $D_2$  anderer Verteilung  $P_2$  übertragen [32, S. 534]. Typischerweise ist der Datensatz von  $P_1$  dabei der deutlich größere [27]. Das Transfer Learning basiert auf der Annahme, dass die für  $D_1$  verwendeten Feature auch bei  $D_2$  anwendbar sind [32, S. 534].

Für die Bildklassifikation kann dies bedeuten, dass die Bilder aus  $D_1$  beispielsweise Katzen und Hunde beinhalten, die Klassifikation für  $D_2$  aber Wespen und Ameisen betrifft [32, S. 534]. Auch für den zweiten Anwendungsfall ist es notwendig, dass das CNN beispielsweise Kanten, Ecken oder Kreise erkennt. Aus Abschnitt 2.4.1 wissen wir, dass die früheren CNN-Schichten allgemeinere Feature erkennen und die tieferen Schichten dann großflächigere Einzelfeature kombiniert betrachten. Daraus lässt sich bereits schließen, dass vor allem die späteren Schichten auf den neuen Anwendungsfall

$D_2$  angepasst werden müssen.

Da der ImageNet-Datensatz wie oben beschrieben eine große Vielzahl und Vielfalt an Bildern beinhaltet, wird dieser häufig als  $D_1$  verwendet und die CNNs darauf vortrainiert (engl. *pretrained*). Transferiert, also übertragen, werden dann die während des Trainings gelernten internen Parameter, die Gewichte des Modells. Man verwendet als Startpunkt für das Modell für  $D_2$  demnach das Endprodukt des Trainings auf  $D_1$ .

Für das Transfer-Learning zur Bildklassifikation lassen sich zwei Ansätze unterscheiden: Das *Fine-tuning* des Modells und das Verwenden des CNNs als *Feature Extractor* [27].

Beim Fine-tuning wird das vortrainierte Modell erneut trainiert, nun allerdings auf den Bildern aus  $D_2$ . Dabei lässt sich noch unterscheiden, ob alle Schichten weiter trainiert werden, oder nur die  $n$  letzten und die vorherigen Gewichte unveränderlich eingefroren werden [52]. Im Allgemeinen ist es zwingend erforderlich, zumindest die letzte Schicht des vortrainierten Netzes auszutauschen und neu zu trainieren, da die Klassenanzahl von  $D_1$  und  $D_2$  meist unterschiedlich ist. Da die Testdaten des ImageNet beispielsweise 1.000 Klassen beinhalten, hat der Modelloutput eines auf ImageNet vortrainiertem Netzwerk auch die Dimension 1.000. Alternativ könnte auch eine weitere Schicht hinzugefügt werden, die als neue Ausgabeschicht fungiert. Die Gewichte der neuen Ausgabeschicht werden zufällig initialisiert [39, S. 423].

Da tiefere Schichten aber Feature enthalten können, die nicht zu dem neuen Anwendungsfall passen, werden auch häufig auch mehr als nur die letzte Schicht trainiert. Dazu werden die früheren Schichten „eingefroren“, sodass deren Parameter unveränderlich sind. Spätere Schichten werden weiter trainiert, jedoch ohne deren Gewichte neu zu initialisieren. Man geht dabei davon aus, dass die trainierten Gewichte einen besseren Ausgangspunkt für das zukünftige Training darstellen [39, S. 424], was zu schnellerer Konvergenz führen kann. Dieses Vorgehen kann auch angewendet werden, um etwaiges Overfitting abzuschwächen [52]. Allerdings lässt sich nicht im Voraus sagen, welcher Anteil des Netzes eingefroren werden soll. Zudem ist zu berücksichtigen, dass der Ressourcenbedarf mit der Anzahl der zu trainierenden Schichten steigt.

Neben dem typischerweise angewendeten Fine-tuning kann das CNN auch als Feature Extractor verwendet werden [52]. Dafür wird die klassifizierende Schicht des Netzes entfernt und der anschließende Netzoutput, der aus den Aktivierungen besteht, als ein Featureset betrachtet, das als Input für einen Klassifizierer wie beispielsweise eine Logistische Regression verwendet wird [52]. Dabei nutzt man, dass die Aktivierungen des CNNs einer Repräsentation der Daten entspricht, die zur Klassifizierung geeignet ist [27]. Dies kann darüber hinaus mit einem Fine-tuning verbunden werden, indem das CNN zunächst auf  $D_2$  trainiert wird und anschließend ohne Klassifizierungsschicht als Feature Extractor genutzt wird [52].

Es gibt aber auch Studien, die dem generellen Anwenden des Fine-tunings von auf ImageNet vortrainierten Netzwerken kritisch gegenüberstehen.

He et al. [53] führten Experimente durch, bei denen sie genau dieses Fine-tuning mit dem Trainieren von zufällig initialisierten Gewichten verglichen. Als Datensatz wurde *Common Objects in Context* (COCO) [54] genutzt. Um den Vorteil der vortrainierten

Netzwerke auszugleichen, wurden die Vergleichsnetzwerke mit mehr Iterationen trainiert.

Die Autoren kamen zu dem Ergebnis, dass die Netze mit zufällig initialisierten Gewichten bezüglich mehrerer Aufgaben und Metriken nicht schlechter abschnitten als das Fine-tuning, auch wenn weniger Trainingsdaten oder komplexere Modelle verwendet wurden [53]. Dies gilt allerdings nur unter dem Vorbehalt, dass mehr Trainingsiterationen verwendet werden und das Vortrainieren führte weiterhin zu schnellerer Konvergenz [53].

Kornblith et al. [55] versuchten herauszufinden, wie gut sich Architekturen, die auf ImageNet erfolgreich sind, auf andere Datensätze übertragen lassen. Dafür verglichen sie 16 Architekturen auf 12 Datensätzen mit Fine-tuning, dem vortrainierten Modell als Feature Extractor und zufällig initialisierten Gewichten (auch als „Training from scratch“ bezeichnet). In der Studie zeigte sich, dass sich die Güte eines Modells zwischen ImageNet als  $D_1$  und weiteren Datensätzen als  $D_2$  gut übertragen lässt. Allerdings kamen sie auch zu dem Ergebnis, dass die Feature, die auf dem ImageNet gelernt wurden, nicht immer gut übertragbar auf andere Datensätze sind. Je größer die Trainingsdatenmenge, Anzahl der Trainingsiterationen sowie der Unterschied zwischen ImageNet und  $D_2$ , desto geringer seien die Vorteile, die das Pretraining mit sich bringt [55].

### 4.3 Transfer Learning für medizinische Bilddaten

Medizinische Bilder unterscheiden sich stark in Farbe, Größe, Inhalt, Position der relevanten Merkmale, etc. von natürlichen Bildern wie sie beispielsweise im ImageNet Datensatz vorkommen. Hinzu kommt, dass die Indikatoren für beispielsweise eine Pneumonie mehrfach in kleineren Bereichen der Lunge auftreten, während die Bilder aus dem ImageNet-Datensatz meistens ein größeres Objekt beinhalten, das über das Label entscheidet [56]. Aus dieser Motivation führten Raghu et al. [56] eine Studie zu Transfer Learning für den RETINA- [57] und den CheXpert-Datensatz [17] durch. Dafür verwendeten die Autoren sowohl eine ResNet- und eine Inception-v3-Architektur [58], aber auch kleinere CNN-Architekturen, die aus den typischen CNN-Bausteinen (Convolution, Pooling, Batch-Normalisierung, ReLU) bestehen. Die Studie zeigte, dass auf dem CheXpert-Datensatz das Fine-tuning nur in einzelnen Fällen signifikante Verbesserungen gebracht hat und die kleineren Netze ähnlich gute Ergebnisse liefern wie die bekannten ResNet und Inception-v3 Modelle. Außerdem führten die Autoren einen Vergleich mit einem sehr kleinen Trainingsdatensatz von 5.000 Bildern durch. Dabei zeigte sich, dass das Pretraining zwar bei der ResNet-Architektur, nicht aber bei den kleineren CNNs zu signifikanten Verbesserungen geführt hat. Dies führen die Autoren darauf zurück, dass das Fine-tuning hauptsächlich den größeren, komplexeren Modellen Vorteile bringt. Des Weiteren testeten die Autoren eine andere Initialisierungsstrategie, die sie als *Mean Var init* bezeichnen und die Training from Scratch und Transfer Learning kombiniert: Statt direkt die vortrainierten Gewichte zu übernehmen, werden nur deren Lageparameter (arithm. Mittelwert  $\bar{x}$  und empirische Varianz  $s^2$ ) ge-

nutzt um für die entsprechende Schicht die Gewichte über Normalverteilung  $N(\bar{x}, s^2)$  zufällig zu initialisieren. So bleibt die Skalierung der vortrainierten Gewichte erhalten, die Features, die sie repräsentieren, werden aber verworfen [56]. Dies führte zu einer relevanten Beschleunigung der Konvergenz. Ein weiteres Experiment von Raghu et al. [56] beschäftigt sich mit einer *weight transfusion*, bei der nur ein Teil der Gewichte für frühere Schichten transferiert und für den Rest des Netzes zufällige Gewichte genutzt werden, sowie das Netz schlanker gemacht wird, indem die Anzahl der Filter halbiert wird. Auch dies führte zu einer beachtlichen Beschleunigung der Konvergenz. Zuletzt testeten die Autoren auch noch eine weitere Initialisierungsstrategie unabhängig von Transfer Learning: die ersten Schichten werden mittels synthetischer Gabor Filter initialisiert und die späteren Schichten zufällig. Dies führte zu einer schnelleren Konvergenz als die vollständig zufällige Initialisierung, war allerdings langsamer als die *weight transfusion*.

Raghu et al. [56] zeigen insgesamt, dass typisches Fine-tuning zwar auch für medizinische Datensätze funktioniert, aber dennoch alternative bzw. Hybrid-Ansätze untersucht werden sollten, da die Vorteile des Transfer Learning auf ImageNet nur beschränkt auf medizinische Daten übertragbar sind.

# Kapitel 5

## Datensätze

Dieses Kapitel stellt einige große Datensätze im Bereich der Thorax-Röntgenbilder vor und begründet die Entscheidung für den CheXpert-Datensatz als Grundlage für die später beschriebenen Experimente.

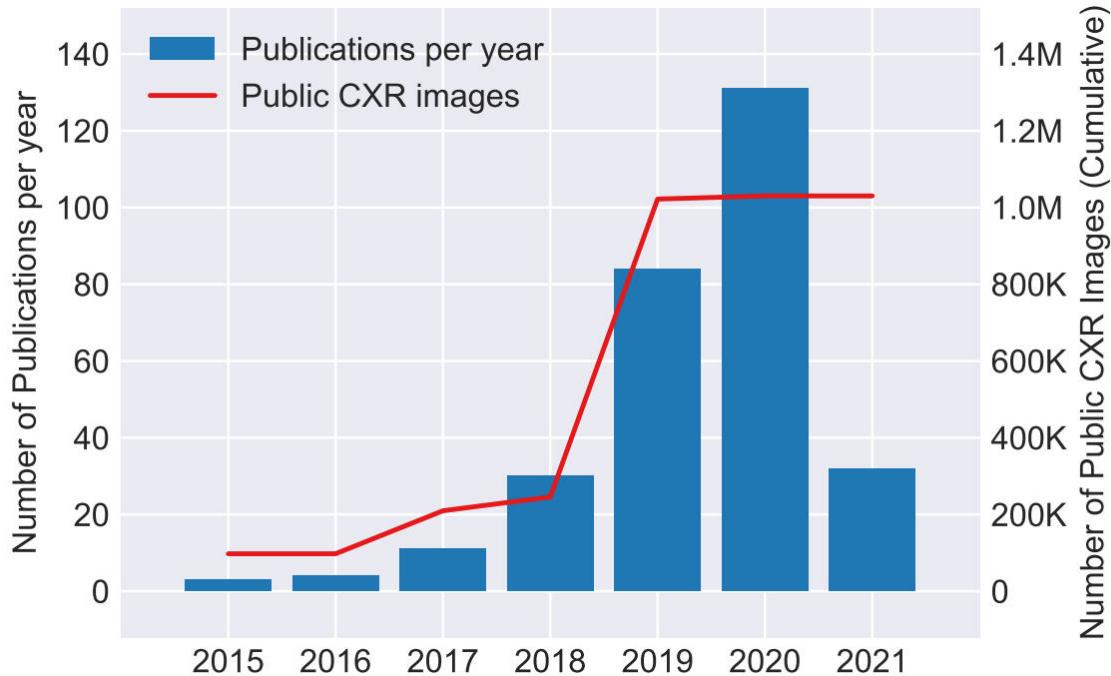
Deep Learning Projekte sind bekannt dafür, dass sie große Datenmengen benötigen, um erfolgreiche Modelle zu erzeugen. Calli et al. [10] führten eine Literaturstudie durch, bei der die positive Korrelation zwischen öffentlich verfügbaren Thorax-Röntgenbildern und dazu veröffentlichten wissenschaftlichen Arbeiten klar ersichtlich wird (siehe Abbildung 5.1).

Bei allen im Folgenden vorgestellten Datensätzen handelt es sich um Multilabel-Datensätze. Es gibt durchaus aber auch Datensätze die sich nur mit einer Krankheit beschäftigen, wie der RSNA Pneumonia Datensatz [59].

### 5.1 ChestX-ray14

Der ChestX-ray14-Datensatz wurde von Wang et al. [60] im Jahr 2017 zunächst als ChestX-ray8 veröffentlicht. Die Zahl steht dabei für die Anzahl der Pathologien. Demnach enthielt er zuerst 8 und wurde dann auf 14 Pathologien zuzüglich des Labels „Normal“ erweitert. Die Originalveröffentlichung spricht von 108.948 Bildern, die von 32.717 Patienten stammen. Der Datensatz, wie er beispielsweise von [61] abgerufen werden kann, beinhaltet 112.120 Bilder von 30.805 Patienten. Es ist unklar, woher diese leichte Diskrepanz kommt. Die Bilder haben eine Auflösung von 1024x1024 und liegen in Graustufen vor [10]. Rund 67.000 Bilder wurden als PA aufgenommen und 45.000 als AP [10].

Die Label wurden über ein NLP-Tool automatisiert aus den Arztberichten erzeugt [60]. ChestX-ray14 war der erste öffentliche Thorax-Röntgenbilder Multilabel Datensatz dieser Größenordnung. Eine Analyse der Label und deren Kookkurrenz wird in Abbildung 5.2 gezeigt.



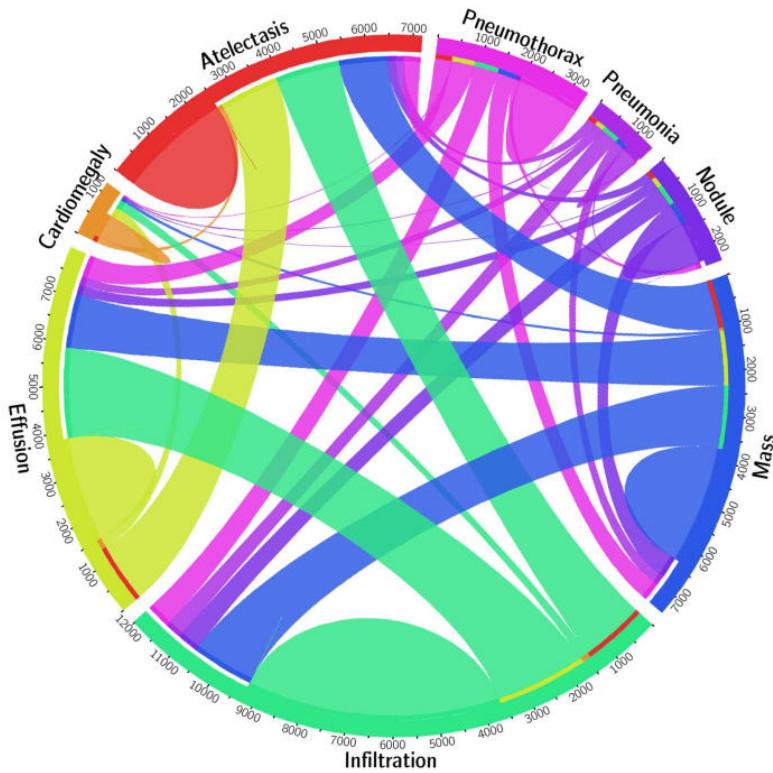
**Abbildung 5.1:** Anzahl der verfügbaren Thorax-Röntgenbilder und der veröffentlichten Arbeiten mit Bezug dazu im Zeitraum 2015 - März 2021. Dies zeigt klar, dass mit der Verfügbarkeit von Daten auch die Anzahl der wissenschaftlichen Arbeiten gestiegen ist. Quelle: [10].

## 5.2 MIMIC-CXR

Der MIMIC-CXR-Datensatz wurde 2019 von Johnson et al.[62] veröffentlicht. Mit rund 377.000 Bildern von etwa 65.000 Patienten ist dies der zum Zeitpunkt der Entstehung dieser Arbeit größte öffentlich verfügbare Datensatz für die Klassifikation von Thorax-Röntgenbildern. Er beinhaltet rund 250.000 frontale und 120.000 laterale Bilder [10]. Diese stammen vom Beth Israel Deaconess Medical Center Emergency Department aus dem Zeitraum zwischen 2011 und 2016 [62]. Die Datensatzautoren nutzten das gleiche Labeling-Vorgehen wie Irvin et al. [17], wodurch sich auch die gleichen Label und Klassen ergeben. Allerdings verwenden sie ein anderes Tool (NegBio [63]). In einer zweiten Version des Datensatzes wurden zusätzlich die Radiologieberichte anonymisiert veröffentlicht [10].

## 5.3 PadChest

Im Jahr 2020 wurde der PadChest Datensatz von Bustos et al. [64] veröffentlicht. Er beinhaltet knapp 160.000 Bilder von rund 70.000 Patienten, die zwischen 2009 und 2017 am spanischen San Juan Hospital aufgenommen wurden. Besonders an diesem Datensatz ist, dass ein großer Anteil, etwa 27%, manuell von Radiologen gelabelt wurde. Mit diesen wurde anschließend ein Rekursives Neuronales Netz trainiert, das für die übrigen Daten die Label erzeugte. Der Datensatz beinhaltet 19 Diagnosen, deren



**Abbildung 5.2:** Analyse der Vorkommen der einzelnen Pathologien und deren Kookkurrenz des ChestX-ray14-Datensatzes in Form eines sogenannten Chord-Diagramms. Dies zeigt, dass beispielsweise die Label Infiltration und Mass häufig gemeinsam mit anderen auftreten, während dies für Pneumothorax eher selten der Fall ist. Außerdem veranschaulicht dies die Unausgeglichenheit zwischen den Klassen, z.B. bei Cardiomegaly. Quelle: [60].

hierarchische Taxonomie dem Unified Medical Language System (UMLS) angepasst wurden [64].

## 5.4 CheXpert

Der CheXpert-Datensatz wurde 2019 von Irvin et al. [17] veröffentlicht. Er enthält 224.316 Bilder von 65.240 Patienten, die im Zeitraum zwischen Oktober 2002 und Juli 2017 am Standford Krankenhaus aufgenommen wurden.

Der Datensatz enthält die folgenden 14 Label: *No Finding, Support Devices, Fracture, Lung Opacity, Edema, Consolidation, Pneumonia, Lesion, Atelectasis, Pleural Other, Pleural Effusion, Pneumothorax, Enlarged Cardiomegaly, Cardiomegaly*. Diese wurden ebenfalls mit einem NLP Tool erzeugt. Dabei ist hervorzuheben, dass das Tool nicht nur positive (1) und negative (0) Label extrahiert, sondern auch ein uncertain (*u* bzw. -1) Label, das dazu dienen soll, die Ungewissheit, die bei den radiologischen Berichten besteht (vgl. Abschnitt 2.1) abzubilden [17]. Ein solches *u* Label kommt bei grob der Hälfte aller Bilder vor, weswegen ihrer Behandlung eine besondere Bedeutung zukommt.

Ein Beispiel für Textpassagen in Radiologieberichten die zu einem *u* Label führen findet

sich bei Johnson et al. [65]. Dabei nutzten die Autoren zwar ein anderes Labeling-Tool, der Text dient aber dennoch als passende Veranschaulichung. So wurde aus dem Satz „Hydropneumothorax previously seen is not as well evaluated on this not fully upright film.“ ein uncertain-Label für Pneumothorax und für „Bibasilar opacities, [...], could be due to atelectasis [...].“ ein uncertain-Label für Atelectasis extrahiert.

Tabelle 5.1 beinhaltet eine Übersicht über die Verteilung der Klassen je Label. Die Werte wurden aus dem heruntergeladenen Datensatz ermittelt und weichen etwas von den originalen Angaben von Irvin et al.[17] ab. Die Ursache dafür ist allerdings nicht festzustellen.

Die Übersicht zeigt, dass der Datensatz zum Teil stark unbalanciert ist. Bei einigen Labels (Lung Opacity, Support Devices, Pleural Effusion und Support Devices) ist die Verteilung der positiven und negativen Klasse nahezu ausgeglichen, ansonsten liegt aber ein Übergewicht der negativen Klasse vor. Dies lässt sich dadurch begründen, dass Support Devices bei mehreren Pathologien auf den Röntgenbildern auftauchen können und so die positive Klasse stärker vertreten ist. Lung Opacity und Pleural Effusion können als Überbegriffe für Symptome verschiedener Krankheiten ebenfalls mit mehreren Labels positiv korrelieren (vgl. auch Abschnitt 5.4.1). Die Unausgeglichenheit der positiven und negativen Klasse ist ansonsten typisch und zu erwarten, wie das folgende Gedankenexperiment verdeutlicht.

Nehmen wir vereinfachend an, wir hätten einen Datensatz mit zehn Pathologien. Wäre dieser ein ausbalancierter Single-Label Multi-Class Datensatz, sollten die Label je 10% positive Beispiele aufweisen. Je näher ein Multi-Label Problem an einer Multi-Class Klassifikation ist, also je geringer die Kardinalität ist, desto näher sollte der Datensatz für die jeweiligen Label eine 10% – 90% Aufteilung aufweisen. Die Kardinalität des CheXpert-Datensatzes hängt von der Behandlung der uncertain-Label ab. Werden sie als positiv interpretiert, beträgt die Kardinalität 2.93, mit Mapping auf die negative Klasse 2.31. Für eine zufällig gezogene Aufnahme sind in diesen Fällen also im Durchschnitt 2.93 bzw. 2.31 der 14 Label positiv.

Die uncertain-Klasse stellt bei den meisten Labels einen eher geringen Anteil (< 6%) dar. Einzig die Label Consolidation und Atelectasis stechen hervor. Bei Consolidation ist der Anteil der uncertain-Label fast doppelt so groß wie der der positiven Klasse und bei Atelectasis sind diese beiden Klassen für je 15% der Daten verantwortlich.

Das Labeling-Tool bewerteten Irvin et al. [17] mittels eines Datensatzes von 1000 Berichten, für den die Radiologen das Labeling händisch vornahmen. Die Resultate zeigten, dass ihr Ansatz deutlich besser funktionierte als das Labeling-Tool des ChestX-ray14-Datensatzes.

Zusätzlich wird produzieren sie eine Baseline. Dabei vergleichen die Autoren fünf Strategien für den Umgang mit den uncertain Labels. Bei *U-Ignore* werden  $u$  Label für die Berechnung des Losses ignoriert. *U-Ones* und *U-Zeros* sind Ersetzungsmethoden, bei denen die uncertain durch die positive respektive negative Klasse ersetzt werden. Die *U-SelfTrained* Strategie betrachtet die  $u$  Label zunächst als ungelabelt und führt ein U-Ignore Training durch. Anschließend werden mit dem so trainierten Modell Vorhersagen erzeugt, um die  $u$  Label zu ersetzen, wenn die Vorhersagen über einem gewissen Schwellwert liegen. Dies wird iterativ wiederholt bis alle uncertain Label

Label	Positiv (%)	Negativ (%)	Uncertain (%)
No Finding	22381 (10.0 %)	201033 (90.0 %)	0 (0.0 %)
Enlarged Cardiomediastinum	10798 (4.8 %)	200213 (89.6 %)	12403 (5.6 %)
Cardiomegaly	27000 (12.1 %)	188327 (84.3 %)	8087 (3.6 %)
Lung Opacity	105581 (47.3 %)	112235 (50.2 %)	5598 (2.5 %)
Lung Lesion	9186 (4.1 %)	212740 (95.2 %)	1488 (0.7 %)
Edema	52246 (23.4 %)	158184 (70.8 %)	12984 (5.8 %)
Consolidation	14783 (6.6 %)	180889 (81.0 %)	27742 (12.4 %)
Pneumonia	6039 (2.7 %)	198605 (88.9 %)	18770 (8.4 %)
Atelectasis	33376 (14.9 %)	156299 (70.0 %)	33739 (15.1 %)
Pneumothorax	19448 (8.7 %)	200821 (89.9 %)	3145 (1.4 %)
Pleural Effusion	86187 (38.6 %)	125599 (56.2 %)	11628 (5.2 %)
Pleural Other	3523 (1.6 %)	217238 (97.2 %)	2653 (1.2 %)
Fracture	9040 (4.0 %)	213732 (95.7 %)	642 (0.3 %)
Support Devices	116001 (51.9 %)	106334 (47.6 %)	1079 (0.5 %)

Tabelle 5.1: Verteilung der Klassen innerhalb der Label des CheXpert-Datensatzes.

ersetzt wurden. Die fünfte Option ist *U-Multiclass*, bei der  $u$  neben 0 und 1 als weitere Klasse betrachtet wird.

Die Autoren erprobten verschiedene Architekturen und entschieden sich schließlich für ein DenseNet-121, das mit Adam für drei Epochen trainiert wurde.

Zur Gütebewertung schlagen die Autoren aufgrund der medizinischen Bedeutung und dem Vorkommen in den (Validierungs-)Daten die Pathologien Atelectasis, Cardiomegaly, Consolidation, Edema und Pleural Effusion vor. Diese Krankheitsbilder werden am Ende dieses Abschnittes kurz beschrieben.

Der Datensatz legt ein Validierungsset von etwa 230 Bildern fest, dessen Label durch einen Mehrheitsentscheid von drei Radiologen erzeugt wurden [17]. Die Autoren nutzten für die Erzeugung der Baseline Ensembling-Techniken, indem sie während dem Training 10 Modell-Checkpunkte speicherten (Horizontal Voting Ensemble, vgl. Abschnitt 2.5) und mit jedem Modell drei Inferenzen durchliefen, wodurch jede Ensemble-Vorhersage als Durchschnitt aus insgesamt 30 Einzelvorhersagen erzeugt wird. Die Autoren nutzten demnach eine 3-fache TTA (vgl. Abschnitt 2.3). Der Vergleich der Strategien ergab, dass auf dem Label Atelectasis die U-Ones Methode und für Cardiomegaly die U-Multiclass Herangehensweise zu den besten AUC-Werten führen. Für die übrigen Vergleichslabel geben die Autoren an, dass sie keinen signifikanter Unterschied feststellen konnten.

Die Datensatzautoren stellen zudem ein geheimes Testset mit 500 Bildern von 500 Patienten zur Verfügung, das ebenfalls von Radiologen gelabelt wurde. Dafür wurden die drei Vorhersagen der Radiologen verwendet und zusätzlich zufällig davon gesampled und anschließend der Mehrheitsentscheid der fünf Vorhersagen gebildet.

Dadurch wird es möglich, einen Vergleich zwischen dem Modell und den Radiologen herzustellen. Die Autoren erreichten mit ihrer Baseline für die Label Atelectasis, Cardiomegaly, Consolidation, Edema und Pleural Effusion einen AUC von 0.85, 0.90, 0.90, 0.92, 0.97. Dies entspricht einem durchschnittlichen AUC von rund 0.91. Das Modell outperformte auf den Labels Cardiomegaly, Edema und Pleural Effusion alle drei Radiologen und auf Consolidation zwei Radiologen, allerdings nicht deren Mehrheitsentscheid. Bei Atelectasis sind alle drei Radiologen besser. Auch bei den Radiologen hat das „Ensembling“ in Form des Mehrheitsentscheids also Verbesserungen gebracht.

Zu dem Datensatz wurde auch eine Competition gestartet, deren bisherige Ergebnisse online unter [15] abrufbar sind. Zum Zeitpunkt der Entstehung dieser Arbeit erreichte der erste Platz der Rangliste einen mittleren AUC von 0.93. Mit Blick auf das Thema der vorliegenden Arbeit ist zudem erwähnens- und beachtenswert, dass die zehn besten Ergebnisse dieser Competition alle von Ensembles der jeweiligen Methode erzielt wurden.

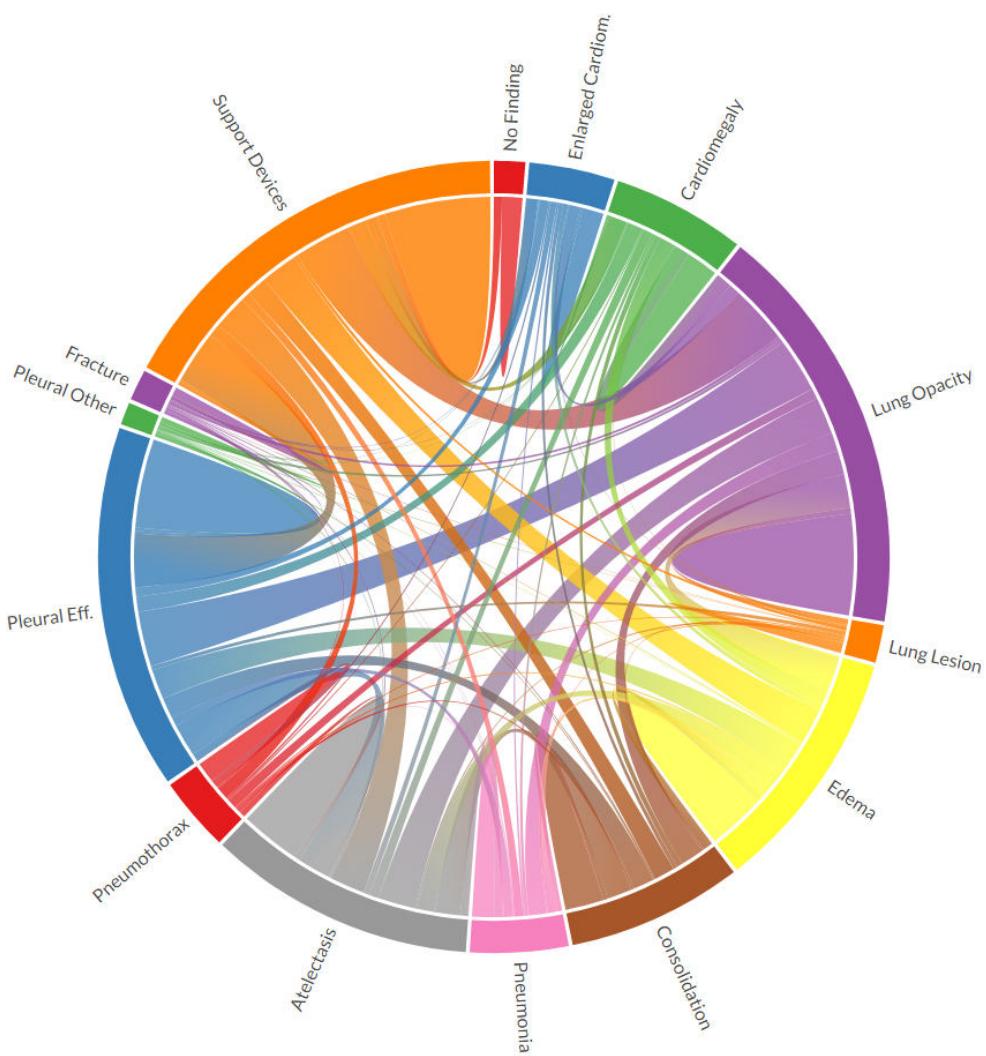
In Anlehnung an Wang et al. [60] lässt sich auch für den CheXpert-Datensatz ein sogenanntes Chord-Diagramm zeichnen, das die Label zirkular anordnet und Kookkurrenzen aufzeigt. Dabei müssen die *u*-Label extra behandelt werden, da für das Chord-Diagramm nur positive und negative Klassen sinnvoll sind. Deshalb wurden zwei Plots erzeugt, wovon für einen der U-Ones Ansatz verwendet wurde (Abbildung 5.3) und für den anderen U-Zeros (Abbildung B.6). Eine interaktive Version der beiden Darstellungen befindet sich auf dem beiliegenden Datenträger.

Diese Diagramme zeigen neben den Kookkurrenzen der Label auch ihre Vorkommenshäufigkeit auf. So lässt sich auf den ersten Blick erkennen, dass zwischen den Klassen eine starke Imbalance herrscht und beispielsweise die Label Fracture, Pleural Other, Lung Lesion und Pneumonia selten auftreten, während Support Devices, Lung Opacity und Pleural Effusion häufig vorkommen.

Weiterhin können die dargestellten Relationen der Label interpretiert werden. So fällt das Label Fracture etwas aus den anderen Pathologien heraus, da man erwarten kann, dass Frakturen nach externer Gewalteinwirkung entstehen, während dies für beispielsweise Infektionskrankheiten und Entzündungen wie der Pneumonie nicht typisch ist. Wie also zu erwarten ist, sind die Kookkurrenzen zwischen dem Label Fracture und den anderen Pathologien gering. Ähnlich lassen sich Schlüsse über das Label No Finding ziehen. Da dieses Label die Absenz der Pathologien kodieren soll, sollten keine Kookkurrenzen vorliegen, was auch der Fall ist. Einzig mit dem Label Support Devices gibt es gemeinsame Vorkommnisse. Dies könnte unter anderem dadurch entstehen, wenn Support Devices auf der Aufnahme sind, die für das Röntgen selbst notwendig waren oder wenn ein Patient nach beispielsweise einer überstandenen Fraktur nun gesund ist, bei der Behandlung aber beispielsweise Knochenschrauben zur Fixierung eingesetzt wurden, die auf der Aufnahme zu sehen sind.

#### 5.4.1 Beschreibung der Krankheitsbilder der Validierungsdaten

Unter *Atelectasis* (dt. Atelektase) versteht man, dass die Lunge oder ein Teil davon kollabiert ist, also eine Verringerung des Lungenvolumens vorliegt [66]. Die Diagnose erfolgt



**Abbildung 5.3:** Chord-Diagramm für den CheXpert-Datensatz mit U-Ones Vorgehen. Die interaktive Version befindet sich auf dem beiliegenden Datenträger.

anhand medizinischer Bildgebungsverfahren (Röntgen, CT), und es lässt sich häufig auch die zugrunde liegende Erkrankung und die Art der Atelectasis bestimmen [66].

Bei einer krankhaften Vergrößerung des Herzens spricht man von einer *Cardiomegaly* (dt. Kardiomegaly, Sportherz) [67]. Dies ist dabei häufig ein Symptom anderer Pathologien [67]. Durch Thorax-Röntgen können anhand des Verhältnisses von Herz und Thorax Indikatoren für eine Cardiomegaly festgestellt werden, wobei auch weitere Diagnoseverfahren eingesetzt werden können [67].

*Consolidation* (dt. Konsolidierung) beschreibt im Bereich der Lunge, dass die Alveolen (Lungenbläschen) mit beispielsweise Flüssigkeiten gefüllt sind [68]. Eine häufige Ursache dafür ist eine Pneumonie [68]. Auf Thorax-Röntgenbildern lässt sich dies anhand weißlich opaken Verschattungen erkennen [68].

Eine *Edema* (dt. Ödem) ist eine abnormale Flüssigkeitsansammlung, die auf beispielsweise eine Entzündung oder Herzkrankheiten zurückgeführt werden kann [66, 69]. Die Diagnose findet üblicherweise anhand von Thorax-Röntgenbildern statt [70].

Diese Kurzvorstellung der Pathologien zeigt bereits, dass obwohl Thorax-Röntgenbilder für die Diagnose essenziell sind, diese jedoch nicht trivial ist, da sich die Indikatoren ähneln oder überschneiden und viele Beziehungen zwischen den Pathologien existieren.

#### 5.4.2 Entscheidung für den Datensatz

Für die vorliegende Arbeit wird hauptsächlich der CheXpert-Datensatz verwendet. Dies liegt darin begründet, dass er durch die uncertain Label zusätzliche Optionen zur Verbesserung der Klassifikation liefert und durch die CheXpert Competition eine kompetitive Forschungsmotivation besteht. Dennoch haben auch die alternativen Datensätze gewisse Vorteile: MIMIC-CXR beinhaltet etwa 150.000 mehr Bilder und der PadChest-Datensatz hat mit knapp 28.000 Bildern die meisten „Gold-Standard Daten“, also solche, deren Label nicht automatisiert, sondern von Menschen erzeugt wurden. Der CheXpert-Datensatz kann sowohl in Originalgröße als auch in der „small“-Version mit verkleinerter Auflösung heruntergeladen werden. Diese Bilder wurden bereits auf eine Größe von  $390 \times 320$  skaliert. Da viele Veröffentlichungen eine Inputgröße von  $224 \times 224$  verwenden, ist dies oft ausreichend.

Da mit steigender Bildgröße auch der Ressourcenbedarf und die Rechenzeit wächst, ist es äußerst unpraktikabel, die Bilder in Originalgröße zu verwenden. Da beim Down-sampling aber Informationen verloren gehen, könnte es sinnvoll sein, auch höhere Auflösungen, wie beispielsweise  $320 \times 320$  oder  $512 \times 512$ , zu verwenden.

Für diese Arbeit wurden beide Varianten des Datensatzes bezogen. Aufgrund der Speicherbeschränkungen der verwendeten Hardware war es allerdings nicht möglich, die Originalbilder entpackt abzuspeichern. Daher wurden diese Bilder bereits beim Entpacken von der Originalgröße auf  $512 \times 512$  herunterskaliert. Ein Training mit höherer Auflösung ist mit der verwendeten Hardware ohnehin nicht vernünftig realisierbar.

# Kapitel 6

## Verwandte Arbeiten

In der Literatur existieren diverse Ansätze zur Verbesserung der Multi-Label Klassifikation von Thorax-Röntgenbildern mittels Methoden der Künstlichen (Visuellen) Intelligenz. In diesem Kapitel werden einige Publikationen vorgestellt, die sich mit der Thematik beschäftigen, für den Implementierungsteil dieser Arbeit aber nur von geringer Bedeutung sind. Weitere Veröffentlichungen, die als Methoden dienten, werden in Kapitel 7 detailliert vorgestellt. Das Kapitel endet mit einer Übersicht über die neuen Entwicklungen des zugrundeliegenden Forschungsbereiches während der COVID-19-Pandemie.

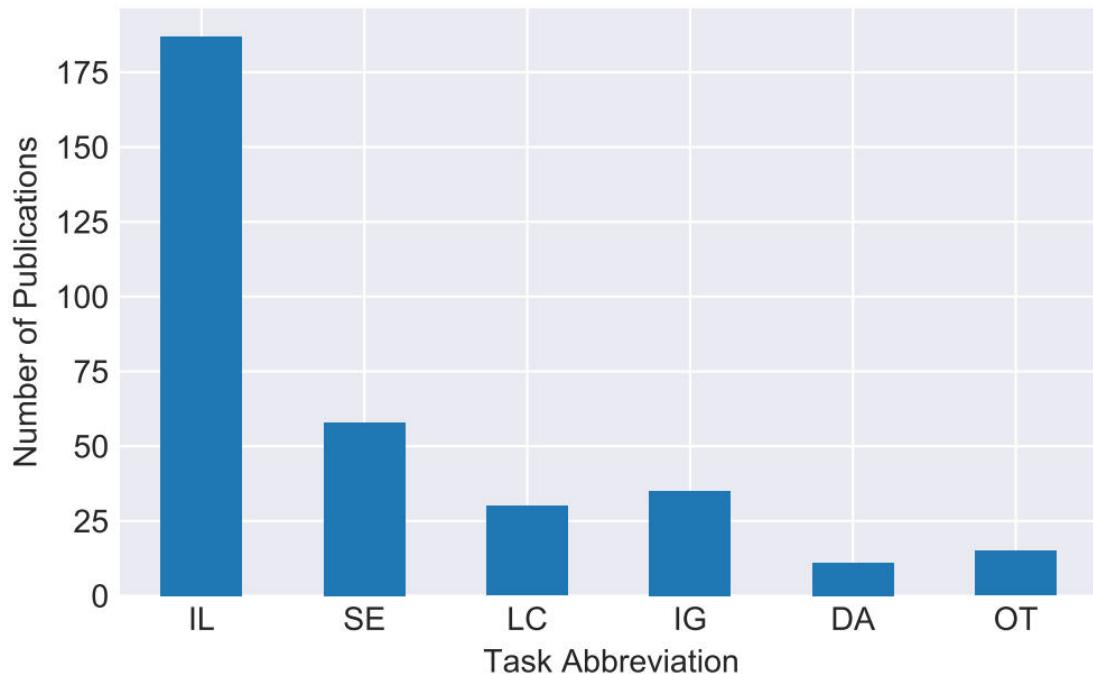
### 6.1 Deep Learning für Thorax-Röntgenbilder

Es gibt mehrere Einsatzmöglichkeiten von Deep-Learning für Thorax-Röntgenbilder. Calli et al. [10] definieren in ihrer Literaturstudie fünf hauptsächliche Anwendungsfälle, wobei die Veröffentlichungen nicht zwingend auf einen davon beschränkt sein müssen. Zuerst ist die *Image-level Prediction* zu nennen. Diese bezeichnet die Klassifikation oder Regression basierend auf einem Röntgenbild. Beispielsweise fiele eine Krankheitsklassifikation oder Regression zur Bestimmung des Schweregrades in diese Kategorie. Die vorliegende Arbeit kann demnach hierin eingeordnet werden.

Der zweite Bereich ist die *Segmentation*, die das Einteilen des Bildes in bestimmte Regionen beziehungsweise Segmente beschreibt. Dies kann unter anderem die Organ-Segmentierung umfassen. Durch die Segmentierung von Herz und Lunge kann zudem der Herz-Lungen-Quotient berechnet werden, der zur Bestimmung einer Kardiomegalie („Sportlerherz“) genutzt werden kann [10].

Mit *Localization* bezeichnen die Autoren die Lokalisierung von Abnormalitäten oder Anatomie (z.B. Rippen) über Bounding Boxes. Die Erzeugung von Heatmaps könnte auch als eine Form der Lokalisierung betrachtet werden. Da diese aber für gewöhnlich nicht nach der Genauigkeit evaluiert wird, wird dies bei der Literaturstudie von Calli et al. [10] nicht zur Lokalisierung gezählt.

Die automatisierte (künstliche) Generierung von Bildern beispielsweise für Bildaugmentierungen oder als Abnormalitätserkennung durch Rekonstruktion ist ein weiterer



**Abbildung 6.1:** Übersicht über die Zuordnung der von Calli et al. [10] untersuchten Veröffentlichungen. Insgesamt wurden dort 295 Studien berücksichtigt. **Abkürzungen:** IL = Image-level Predictions, SE = Segmentation, LC = Localization, IG = Image Generation, DA = Domain Adaptation, OT = Other. Quelle: [10].

Anwendungsfall, den die Autoren *Image Generation* nennen.

Zuletzt wird in der Publikation noch die Kategorie *Domain Adaption* beschrieben. Diese beinhaltet Untersuchungen der Generalisierungsfähigkeit eines Modells durch Evaluation auf einem externen Datensatz, also Bildern aus einer anderen Domäne bzw. einer anderen Verteilung. Dies geschieht in dieser Arbeit in Kapitel 12.

In Abbildung 6.1 befindet sich eine Übersicht über die Anzahl der Veröffentlichungen je Kategorie. Diese zeigt, dass die Image-level Prediction stark dominiert, gefolgt von Segmentation und Image Generation. Dabei sollte berücksichtigt werden, dass auch hier eine Korrelation zwischen den verfügbaren Datensätzen und den Publikationen zu erwarten ist (vgl. Abbildung 5.1).

### 6.1.1 Automated abnormality classification of chest radiographs using deep convolutional neural networks

Tang et al. [71] stellen einen Vergleich verschiedener bekannter CNN-Architekturen auf dem ChestX-ray14-Datensatz [60] an. Dabei evaluieren sie für die Architekturen AlexNet, VGG16/19, ResNet-18/-50, Inception-v3 und DenseNet-121 jeweils ein vortrainiertes Modell und eines, bei dem die Gewichte zufällig initialisiert wurden. Die Autoren führten dabei allerdings keine Multi-Label Klassifikation durch, sondern lediglich eine binäre Unterscheidung zwischen normal (eines der Label) und abnormal

(alle anderen Label). Neben dem Vergleich der Modelle und dem Effekt von Transfer Learning, verglichen die Autoren nicht nur Bilder der Größe 224x224, sondern auch die Auflösungen 512x512 und 1024x1024. Ferner führten sie eine externe Validierung durch, um den Einfluss einer Domänenverschiebung zu erörtern.

## Resultate

Die Modelle erzielten allesamt sehr gute Resultate mit einem AUC von über 0.96. Die vortrainierten Modelle schnitten dabei immer leicht besser ab als die zufällig initialisierten. Das Transfer Learning hat sich also als nützlich erwiesen. Höhere Auflösungen führten hingegen nicht zu signifikant besseren Ergebnissen. Bei der Untersuchung der Domänenadaptation führten die Autoren Experimente mit zwei externen Datensätzen durch: Dem Open-I-Datensatz [72] und dem WCMC-Datensatz [73], der ausschließlich Röntgenaufnahmen von Kindern enthält und somit eine andere Kohorte abbildet. Das auf dem ChestX-ray14 trainierte Modell performte auf den Open-I Daten hinreichend gut und ein Fine-tuning auf dem Datensatz brachte keine Verbesserungen. Wohingegen bei dem pädiatrischen Datensatz das Fine-tuning auf den Bildern der fremden Kohorte eine Verbesserung der AUC von 0.916 auf 0.975 brachte. Die Autoren schließen daraus, dass Fine-tuning auf einem externen Datensatz dann vorteilhaft ist, wenn größere Unterschiede den Domänen oder Kohorten vorliegt.

### 6.1.2 Assessment of convolutional neural networks for automated classification of chest radiographs

Dunnmon et al. [74] führen ebenfalls eine Klassifikation nach normal – abnormal durch. Die Autoren verglichen mehrere typische CNN-Architekturen wie AlexNet, ResNet und DenseNet, sowohl vortrainiert als auch zufällig initialisiert.

Neben der Skalierung auf  $224 \times 224$  und der Normalisierung der Bilder, wendeten sie zusätzlich eine sogenannte *Histogram Equalization (HE)* auf die Bilder an, um den Kontrast zu verbessern [74]. Die HE gleicht die Verteilung der Graustufen eines Bildes an. Es gibt mehrere Varianten und Erweiterungen dieses Algorithmus. In der einfachsten Form, der sogenannten *Global Histogram Equalization* wird dafür das gesamte Bild auf einmal betrachtet [75]. Sie wird berechnet, indem im ersten Schritt die kumulative Dichtefunktion als Summe der Auftrittswahrscheinlichkeiten eines Graustufenwertes gebildet wird [75]. Für ein Bild mit dem Wertebereich der Graustufen  $[0, L - 1]$  – meistens gilt  $L = 256$  – kann für den Graustufenwert  $r_i$  mit  $k = 0, 1, \dots, L - 1$  dessen Auftrittsfrequenz berechnet werden als

$$P(r_i) = \sum_{i=0}^k \frac{n_i}{n} \quad (6.1)$$

wobei  $n$  die Anzahl aller Pixel und  $n_i$  die Anzahl der Pixel des Graustufenwertes  $r_i$  ist [75]. Damit lässt sich mit

$$C(r_k) = \sum_{i=0}^k P(r_i) \quad (6.2)$$

die kumulative Dichtefunktion  $C(r_k)$  berechnen [75].

Ein Beispiel für die Auswirkungen der HE befindet sich in Abbildung 8.4 auf Seite 93.

Die Autoren nutzten einen nicht-öffentlichen Datensatz, der aus dem Zeitraum 1998 bis 2012 stammt und insgesamt knapp 216.000 frontale Thorax-Röntgenbilder enthält. Sie kamen zu dem Ergebnis, dass bereits ein Datensatz der Größenordnung 20.000 Bilder ausreichend ist, um einen Klassifizierer zu trainieren, der gute Ergebnisse für die Klassifikationsaufgabe normal – abnormal erzeugt.

Die Label für dieses Trainingsset können dabei durchaus nur von einem Radiologen erzeugt worden sein, da ein gewisser Label-Noise nicht zu Verschlechterungen führt. Wichtiger sei es, dass die Label des Evaluationsdatensatzes absolut korrekt sind.

### 6.1.3 Weakly Supervised Lesion Localization With Probabilistic-CAM Pooling

Ye et al. [76] stellen die *Probabilistic Class Activation Map (PCAM)* vor, eine neue Methode zur Lokalisierung von Läsionsbereichen (engl. *lesion area*) in Thorax-Röntgenbildern, die gleichzeitig die Klassifikation verbessern soll.

Derartige Bildregionen können beispielsweise als Heatmaps dargestellt werden und sind für einen etwaigen Praxiseinsatz sinnvoll, da sie einerseits der Modell-Klassifizierung Erklärbarkeit verleihen und andererseits die Radiologen auf relevante Bildbereiche aufmerksam machen können.

Sollen solche Heatmaps akkurat erstellt werden, ist es allerdings für gewöhnlich nötig, dies überwacht zu trainieren [76]. Dafür müsste in den Trainingsdaten jedoch die Information über den korrekten Läsionsbereich vorliegen. Dies trifft bei den öffentlichen Thorax-Röntgenbilder Datensätzen nur auf einen geringen Anteil der Bilder zu. Denn während die Krankheitslabel mittels NLP-Tools aus den Radiologieberichten erzeugt werden können, ist dies für die Läsionsbereiche nicht möglich. Die dafür nötigen Angaben sind nur grob textuell erfasst, beispielsweise könnte der Text die Passage „right rib fractures“ [17] enthalten. Dies entspricht allerdings nicht der benötigten Genauigkeit, die bestenfalls in Pixelangaben vorliegen sollte. Demnach wäre eine manuelle Annotation durch Radiologen unabdingbar, was zu hohen Kosten führen würde.

Eine Möglichkeit, um dennoch Heatmaps zu erzeugen, besteht in dem sogenannten *Class Activation Mapping (CAM)* von Zhou et al. [77]. Dieses basiert darauf, dass CNNs in der Lage sind, Objekte auch unüberwacht (also ohne derartige Annotation) zu lokalisieren, diese Information aber durch das Nutzen einer FC-Schicht zur Klassifikation verloren geht.

Für das CAM wird vor der FC-Schicht ein Global Average Pooling verwendet. Durch die Gewichtungen der Output-Schicht wird die Bedeutung der letzten Feature Maps und somit der einzelnen Bildregionen bestimmt. Dafür wird eine gewichtete Summe der Feature Maps je Klasse gebildet.

Dieses Prinzip kann auch für Thorax-Röntgenbilder angewendet werden und wird von Ye et al. [76] genutzt, um das Training zu verbessern.

Dafür wird zunächst der Forward-Pass mit einem Convolutional Backbone durchgeführt. Anschließend werden für ein Label die Feature Maps einzeln in eine FC-

Schicht gegeben, die als  $1 \times 1$  Convolution implementiert ist, und auf das eine Sigmoid-Aktivierung folgt. Durch die Sigmoid-Aktivierung kann der Output als Wahrscheinlichkeit interpretiert werden, dass das gewählte Label positiv ist. Daher ergibt sich eine probabilistische CAM, die PCAM, die für die Erzeugung einer Heatmap des Labels genutzt werden kann.

In einem nächsten Schritt wird diese PCAM normalisiert und als Gewichtung für die Kombination der ursprünglichen Feature Maps in Form eines gewichteten Average Poolings verwendet. Daraus ergibt sich ein Feature Embedding, das erneut in die FC-Schicht gegeben wird, um für das gewählte Label eine Vorhersage zu erzeugen.

Der zentrale Unterschied zwischen PCAM und CAM besteht also in der Sigmoid-Aktivierung und der Verwendung der Aktivierungsmap für das Training. Ye et al. [76] erreichten auf dem ChestX-ray14-Datensatz durch die PCAM-Methode sowohl für die Lokalisierung als auch für die Klassifizierung eines Großteils der Label AUC-Werte, die den Stand der Technik verbesserten. Bei der CheXpert Competition belegen die Autoren mit ihrer Methode zum Zeitpunkt der Entstehung dieser Arbeit den fünften Platz. Unter [78] steht eine Implementierung der Methode zur Verfügung.

#### **6.1.4 When does bone suppression and lung field segmentation improve Chest X-Ray disease classification**

Baltruschat et al. [16] führten eine Studie zum Effekt von zwei komplexeren Bildvorverarbeitungsschritten, der *Bone suppression* und *Lung field segmentation* durch.

##### **Bone suppression**

Auf den Thorax-Röntgenbildern ist neben der Lunge auch das Thorax-Skelett sichtbar und die Rippen sind klar im Lungenbereich erkennbar. Jedoch enthalten die Knochen – solange keine Fraktur erkannt werden soll – keine relevanten Informationen und könnten das Klassifizierungsergebnis durch einen Noise-Effekt beziehungsweise Verschattungen verschlechtern.

Die Methode der Bone suppression, bei der die Rippen auf der Röntgenaufnahme (nahezu) unsichtbar gemacht werden, wurde ursprünglich für Radiologen entwickelt und der Einsatz führte in einer Studie [79] zur Verbesserung der Krankheitserkennung, weswegen es nahe liegt, diesen Vorverarbeitungsschritt auch für das Training von CNNs zu evaluieren. Baltruschat et al. [16] nutzen dafür ein ähnliches Vorgehen wie von Berg et al. [80], das auf der ST-Transformation beruht. Es existieren aber auch Deep-Learning-basierte Ansätze, siehe beispielsweise [81].

##### **Lung field segmentation**

Mit der Lung field segmentation soll aus dem Röntgenbild der Lungenbereich ausgeschnitten werden. Damit wollen die Autoren den Informationsverlust reduzieren, wenn die Auflösung des Bildes für den Netz-Input verkleinert wird. Dafür nutzten sie zunächst ein *foveal* CNN nach Brosch et al. [82]. Dessen Output wurde weiter verarbeitet, indem eine Bounding Box um die beiden größten verbundenen Regionen mit zusätzlichem Padding als Lungenbereich ausgeschnitten wurde.

## Resultate

Die beiden Vorverarbeitungen führten jeweils zu leichten Verbesserungen, wobei der Einfluss der Lung field segmentation stärker war. Die Kombination beider Schritte führte im Vergleich zur ausschließlichen Lungensegmentierung nur zu marginalen Verbesserungen. Werden jedoch Ensembles verglichen, ist der Vorteil durch die Vorverarbeitungen am größten.

Die Autoren begründen dies dadurch, dass die bearbeiteten Bilder zu unterschiedlichen lokalen Optima führen, was durch einen Vergleich der Korrelation der einzelnen Ensemble-Modelle bekräftigt wird. Demnach scheint diese Bildvorverarbeitung besonders beim Einsatz von Ensembling vorteilhaft zu sein.

### 6.1.5 Zusammenfassung dieser Resultate

Zusammenfassend zeigen diese Veröffentlichungen, dass Transfer Learning und Finetunen auf die konkreten Gegebenheiten des Einsatzbereiches einen positiven Effekt zu haben scheint, vor allem wenn zwischen dem ursprünglichen Trainingsdatensatz und der Kohorte, in der das Modell eingesetzt werden soll, größere Unterschiede bestehen, sodass eine Adaption erforderlich ist. Des Weiteren muss ein gewisser Grad an Rauschen der Label nicht zwingend schädlich für die Modell-Güte sein – zumindest wenn er nur um die Klassifikation abnormaler Thorax-Röntgenbilder geht. Die Histogram-Equalization kann als zusätzliche Bildaugmentierung verwendet werden, wodurch Feature besser erkennbar gemacht werden können. Dahingegen kann nicht pauschal davon ausgegangen werden, dass eine höhere Bildauflösung zu besseren Resultaten führt.

## 6.2 Entwicklungen während der COVID-19 Pandemie

Da mit COVID-19 eine neue Lungenkrankheit aufgetreten ist, die sich während des Jahres 2020 zur globalen Pandemie entwickelte, und in den Jahren davor das Forschungsinteresse an Deep Learning für Thorax-Röntgenbilder stark gewachsen ist, ist eine logische Schlussfolgerung, dass sich die Forschung auch mit dem Einsatz von Deep Learning für COVID-19 beschäftigt. Zu den Anwendungsbereichen zählt unter anderem die Vorhersage des lokalen Infektionsgeschehens [83], die Beurteilung des Schweregrades [84] und die Detektion der Krankheit auf CT [85] oder Röntgenbildern [86, 87, 88].

Der zum Zeitpunkt der Entstehung dieser Arbeit größte verfügbare Datensatz mit COVID-19 Röntgenbildern wurde von Cohen et al. [89] veröffentlicht und unter [42] zur Verfügung gestellt. Dieser enthält zum Zeitpunkt der Entstehung dieser Arbeit 468 COVID-19 Bilder, wovon einige allerdings auch CT-Aufnahmen sind. Damit ist die Anzahl der öffentlich verfügbaren Bilder sehr gering. Obgleich viele Veröffentlichungen sehr gute Metriken berichten – beispielsweise wurde von Minaee et al. [87] eine AUC von mehr als 0.97 mit vier verschiedenen Architekturen erreicht – sind diese Ansätze kritisch zu betrachten. Viele Veröffentlichungen kombinieren die positiven COVID-19 Bilder mit Bildern eines anderen Datensatzes als negative Klasse um das

binäre Klassifikationsproblem der COVID-19 Erkennung zu formulieren [90]. Die Generalisierungsfähigkeit derartiger Netze erwies sich als äußerst schlecht, da sie nicht lernen, COVID-19 zu erkennen, sondern die Datensätze zu unterscheiden [90]. Aufgrund dieser Defizite ist eine Klassifizierung von COVID-19 anhand der aktuell öffentlich verfügbaren Daten nicht sinnvoll, besonders für einen potenziellen Praxiseinsatz. Allerdings ist es wahrscheinlich, dass sich die Datenlage in der Zukunft verbessern wird und somit derartige Studien sinnvoller werden.

# Kapitel 7

## Ausgewählte Methoden

Ziel dieser Arbeit ist es, mehrere Ansätze aus der Literatur zur Multi-Label Klassifikation von Thorax-Röntgenbildern unter Nutzung von Ensembling-Techniken zu kombinieren.

Im Folgenden Kapitel werden die Methoden vorgestellt, die zur Implementierung und als Teil des Ensembles ausgewählt wurden. Dabei wird auf die Besonderheiten der jeweiligen Veröffentlichung eingegangen und die Auswahl begründet. Mit Hinblick auf das Thema der vorliegenden Arbeit, werden zudem etwaige Ensembling-Methoden der Veröffentlichungen benannt.

Die Beschreibung der Implementierung der einzelnen Methoden findet in Kapitel 8 statt.

### 7.1 FRODO: Free rejection of out-of-distribution samples: application to chest x-ray analysis

Calli et al. [91] stellen eine Out-of-Distribution (OoD) Erkennung für Thorax-Röntgenbilder vor, die *free rejection of out-of-distribution* (FRODO). Ihren Ansatz bezeichnen sie als gratis (engl. *free*), da sie ein zur Krankheitsklassifizierung trainiertes CNN verwenden und keine OoD-Trainingsdaten benötigt werden.

Die Methodik besteht zunächst darin, die erwünschte Distribution zu definieren, was über die Logits innerhalb des Netzwerkes realisiert wurde. Soll dann über die Zugehörigkeit eines Bildes zur Distribution entschieden werden, wird ein Forward-Pass durchgeführt und dann die Mahalanobis-Distanz zwischen den so erzeugten Logits und der vorher festgelegten Ziel-Distribution berechnet. Die Entscheidung, ob das Bild OoD ist oder nicht, wird also anhand eines Abstandsmaßes gefällt. Dementsprechend muss ein Schwellwert definiert werden, ab dem der Netz-Input als OoD deklariert werden soll. Dazu geben die Autoren allerdings keine genaueren Informationen.

Dieses Vorgehen bekräftigt die Behauptung, dass der Ansatz gratis ist. Gehen wir von einem Szenario aus, in dem ein Computer-aided diagnosis (CAD)-System zur Klassifikation von frontalen Thorax-Röntgenbildern eingesetzt wird. Möchte ein An-

wender nun ein laterales Thorax-Röntgenbilder klassifizieren und verwendet dieses als Input für das System, würde intern ein Forward-Pass durchgeführt. So werden zwar Vorhersagen erzeugt, jedoch sollte die OoD-Detektion anschlagen und dem Anwender könnte statt den Prognosen eine Warnung gezeigt werden, die darauf hinweist, dass das System nicht in der Lage ist, derartige Bilder zu verarbeiten. Ohne eine solche OoD Detektion würde das CAD-Tool dem Nutzer eine invalide Diagnose suggerieren. Um die Absurdität dessen aufzuzeigen, lässt sich auch beispielsweise ein Katzenbild als Input verwenden. Auch für dieses würde das Netz Vorhersagen erzeugen, jedoch mit undefiniertem Ausgang.

Diese Gedankenexperimente zeigen auf, dass eine OoD-Detektion für einen etwaigen Praxiseinsatz unabdingbar ist.

Die von den Autoren genutzte Mahalanobis-Distanz wurde von Mahalanobis et al. [92] als „generalisierte Distanz“ vorgestellt. Sie lässt sich berechnen als

$$\text{MD}(x_i) = \sqrt{(x_i - \bar{x}) C_x^{-1} (x_i - \bar{x})^T}, \quad (7.1)$$

wobei  $x$  die Beispiele sind,  $x_i$  das Objekt dessen Distanz berechnet werden soll,  $C_x^{-1}$  die inverse Kovarianzmatrix von  $x$  ist und  $\bar{x}$  der arithm. Mittelwert von  $x$  ist [93].

Als Backbone verwenden die Autoren ein auf ImageNet vortrainiertes ResNet-50, das sie auf einem Subset des ChestX-ray14-Datensatzes Fine-tunen. Dieser enthält ausschließlich frontale Röntgenbilder. Die Autoren führen die Experimente für verschiedene Schichten des ResNets durch. Genauer gesagt für die letzte Convolution-Schicht von jedem der fünf Convolutional-Blöcke (vgl. Abbildung 2.8).

Die Autoren kamen zu dem Ergebnis, dass die Aktivierungen aus dem Block „conv3“ (vgl. Abbildung 2.8) die beste OoD-Erkennung ermöglichen. Sie schlussfolgern daraus, dass wenn man das Netzwerk als Encoder-Decoder Paar betrachtet, conv3 die tiefste Encoding Schicht darstellt und somit die Features erzeugt, die die Daten am besten beschreiben. Die Autoren Calli et al. [91] erzielten in ihren Experimenten einen AUC von 0.99 für die Erkennung von OoD-Bildern, während sie mit der Baseline-Methode von Hendrychs et al. [94], welche die Outputs der Softmax-Funktion zur Bestimmung der Distribution-Zugehörigkeit verwendet, einen AUC von 0.8 erreichten. Die größten Schwierigkeiten hatte die FRODO-Methode mit lateralen Bildern, diese wurden jedoch zu 90% korrekterweise als OoD erkannt.

Da für die zugrundeliegende Arbeit der CheXpert-Datensatz genutzt wird und für diesen auch lateral aufgenommene Bilder Teil der Distribution sein sollen, lässt sich das Vorgehen für diesen Fall nicht direkt imitieren. Stattdessen muss um die korrekte Distribution festzulegen ein Datensatz verwendet werden, der sowohl frontale als auch laterale Aufnahmen und möglichst keine Bilder enthält, deren Qualität als zu schlecht empfunden wird.

### 7.1.1 Nutzung von Ensembling-Methoden

In der Arbeit von Calli et al. [91] und bei dessen Implementierung in der vorliegenden Arbeit wurden keine Ensembling-Methoden genutzt. Es wäre jedoch prinzipiell denkbar, auch auf die OoD-Detektion ein Ensembling analog zur Klassifikation anzuwenden. Darüber hinaus könnte auch eine TTA eingesetzt werden. Dabei sollte aber berücksichtigt werden, dass die angewendeten Bildaugmentierungen nicht nur das Label bewahren, sondern auch die Zugehörigkeit zur gewünschten Distribution nicht beeinflussen.

## 7.2 Automated Triaging of Adult Chest Radiographs with Deep Artificial Neural Networks

Annarumma et al. [11] stellen eine Studie zum Einsatz von Deep Vision für die automatisierte Einordnung von Thorax-Röntgenbildern nach deren Dringlichkeit vor.

### 7.2.1 Datensatz, Label, Bildaugmentierung

Der (nicht öffentliche) Datensatz der Autoren besteht aus etwa 830.000 Bildern, die von drei Krankenhäusern zwischen Januar 2005 und Mai 2017 aufgenommen wurden. Von diesen Bildern waren allerdings nur ca. 680.000 im passenden Format. Außerdem entschieden sich die Autoren dazu, die Bilder von Patienten unter 16 Jahren zu entfernen, wodurch sich der tatsächlich verwendete Datensatz auf etwa 470.000 Bilder reduzierte.

Selbst mit diesen Reduktionen ist der verwendete Datensatz noch deutlich größer als alle zum Zeitpunkt der Entstehung dieser Arbeit öffentlich verfügbaren Datensätze (vgl. Kapitel 5) und beinhaltet mehr als doppelt soviel Bilder wie der in der vorliegenden Arbeit genutzte CheXpert-Datensatz (ca. 220.000 Bilder).

Die Thorax-Röntgenbilder wurden unter Berücksichtigung des Patientenalters und der Pathologien stratifiziert auf einen Trainings- (80% der Daten), Validierungs- (10%) und Testdatensatz (10%) aufgeteilt.

Mittels eines NLP-Tools wurden aus den Röntgenberichten 15+1 Label extrahiert, wovon eines „Normal“ bzw. „No finding“ ist. Diese Label wurden dann nach der Dringlichkeit kategorisiert als entweder

- kritisch (engl. *critical*): Unverzügliches Handeln, z.B. Pneumothorax oder,
- dringend (engl. *urgent*): Handeln innerhalb von 48 Stunden, z.B. Consolidation oder,
- nicht dringend (engl. *nonurgent*): Handeln im Zeitraum des typischen Praxisalltags, z.B. Hiatus Hernia oder,
- normal (engl. *normal*): Kein Pathologie erkannt.

Die einzelnen Pathologien sowie deren Dringlichkeit sind in Abbildung 7.1 aufgelistet.

Die Qualität des NLP-Tools wurde anhand eines Teildatensatzes mit rund 4.500 Bildern beurteilt, dessen Label von zwei Radiologen manuell validiert wurden, und wird von den Autoren als „very good“ eingestuft [11]. Der F1-Score für die Kategorien normal / nonurgent / urgent / critical betrug 0.97 / 0.88 / 0.95 / 0.90 [11].

**Table 1: List of Selected Radiologic Labels and Corresponding Priority Levels**

Radiologic Label	Priority Level
Abnormal-other	Nonurgent
Airspace opacification/consolidation	Urgent
Bone lesion/abnormality	Urgent
Cardiomegaly	Nonurgent
Collapse	Urgent
Hiatus hernia	Nonurgent
Interstitial shadowing	Urgent
Intra-abdominal pathology	Critical
Medical device	Nonurgent
Paratracheal/hilar enlargement	Urgent
Parenchymal lesion	Urgent
Pleural effusion/abnormality	Urgent
Pneumomediastinum	Critical
Pneumothorax	Critical
Subcutaneous emphysema	Critical

Note.—An additional “normal” label was used for radiographs with no abnormalities.

**Abbildung 7.1:** Pathologie und zugehöriges Prioritätslevel, wie sie von Annarumma et al. [11] verwendet wurden. Quelle: [11].

Vor dem Training führten die Autoren noch einen Vorverarbeitungsschritt durch. Da besonders die Thorax-Röntgenbilder aus der Intensivstation Bildartefakte aufwiesen, wie z.B. Textinformationen über das verwendete Röntgengerät, nutzten Annarumma et al. [11] einen Objekterkennungsalgorithmus, um diese mit schwarzen Boxen zu überdecken und einen möglichen Bias zu verhindern. Im Zuge dessen wurden auch auf Bildern ohne derartige Artefakte an den Rändern schwarze Boxen eingefügt. Damit soll verhindert werden, dass das Modell lernt, Bilder aus der Intensivstation anhand der Artefakte zu erkennen.

Zusätzlich wurden die Bildaugmentierungen Zuschneiden, Padding und Rotation angewendet, um den Datensatz zu erweitern.

Mittels des so formulierten Datensatzes wurden zwei CNNs *end-to-end* trainiert, um die Vorhersage der Dringlichkeit zu automatisieren [11].

## 7.2.2 Prioritätsprognose

Für die Prognose haben die Autoren ein Ensemble aus zwei CNNs der Inception-v3-Architektur verwendet. Die Netze wurden auf Bildern unterschiedlicher Auflösung trainiert. Eines auf Bildern der Größe  $1211 \times 1083$  und das andere auf  $299 \times 299$ , was der

typischen Auflösung für die Inception-v3 entspricht. Die Ausgaben der beiden CNNs wurden gemittelt, um die Ensemble-Vorhersage zu erhalten.

Die Autoren stellten in ihren Experimenten fest, dass ein Ensemble aus zwei CNNs unterschiedlicher Bildauflösung bessere Ergebnisse liefert als die beiden einzelnen Modelle und auch als ein einzelnes Netz, das noch größere Auflösungen verwendet. Dies führen sie darauf zurück, dass Pathologien wie Cardiomegaly, die sich in einem größeren Lungenbereich identifizieren lassen, gut auf kleineren Bildern erkannt werden können. Sie weisen aber darauf hin, dass derartiges Downsampling bei medizinischen Bildern Risiken birgt, da wichtige Features verloren gehen könnten. Für diese Abwägung muss zusätzlich der Rechen- und Speicheraufwand berücksichtigt werden. Auch sollte in Betracht gezogen werden, dass selbst die kleinere der beiden verwendeten Auflösungen etwa 20% größer ist als die typische Eingabegröße von  $224 \times 224$ , wie sie häufig für Architekturen wie das DenseNet oder ResNet verwendet wird.

Die Autoren führten zunächst eine Multi-Label Klassifikation durch, indem sie die Vorhersage der 15 Pathologien je als binäre Klassifikation nach dem Vorhandensein des jeweiligen Labels vornahmen. Sie transformierten die Multi-Label Klassifikation also in ein Binary Relevance One vs. Rest Problem (vgl. Kapitel 3.2). Anschließend wurde die Vorhersage der Priorität als ein ordinales Regressionsproblem modelliert. Dies bietet sich an, da die Dringlichkeitsstufen eine intrinsische Ordnung haben: normal entspricht dem Wert 1 und kritisch der Stufe 4. Die Autoren trainierten dann drei binäre Klassifikatoren – die jeweils ein Ensemble sind –, von denen jeder  $k$ -te entscheiden sollte, ob auf dem Bild eine größere Dringlichkeit als  $k$  vorliegt oder nicht. Die Priorität normal entspricht dann einer negativen Vorhersage aller drei Ensembles und die kritische Priorität einer dreifach positiven Prognose.

Jeder der drei Klassifikatoren bestand aus einer einzelnen FC-Schicht mit zwei Ausgabeneuronen. Als Input dafür wurden die Ausgaben der letzten AveragePooling-Schicht genutzt. Somit wurde hier eine Form des Transfer Learning angewendet, indem die CNNs als Feature Extractor dienen (vgl. Kapitel 4.2).

Zusammengefasst setzen Annarumma et al. [11] eine Problemtransformation, Ensembling und Transfer Learning sowie spezifisches Domänenwissen ein, um die Dringlichkeit eines Thorax-Röntgenbildes zu bestimmen. Auch wenn das Ziel dieser Arbeit nicht in der Priorisierung, sondern in der Krankheitsklassifikation liegt, sind die Erkenntnisse bezüglich dem positiven Effekt des Ensembling und dem Einfluss der Bildauflösung wichtig für die vorliegende Problemstellung. Die Erkenntnisse bezüglich dem Einfluss von Bild-Artefakten sind allerdings nicht übertragbar, da nicht klar ist, ob bei dem verwendeten CheXpert-Datensatz ebenfalls ein Bias bezüglich der Intensivstation vorliegt. Dies betont aber die Bedeutung der Domänenexpertise für etwaige Praxiseinsätze.

### 7.2.3 Nutzung von Ensembling-Methoden

Annarumma et al. [11] nutzen Ensembling-Techniken, indem sie zwei Modelle mit unterschiedlichen Bildauflösungen trainieren und für die Vorhersage den Mittelwert der beiden Modelle bilden. Dies wirkte sich positiv auf die Resultate aus und führte zu besseren Ergebnissen als die Modelle einzeln erreichten.

### 7.2.4 Resultate

Die Genauigkeit der Priorisierung bezeichnen die Autoren als „good“ [11]. Der F1-Score für normal / nonurgent / urgent / critical betrug 0.72 / 0.45 / 0.78 / 0.63.

Zusätzlich wurde eine Simulationsstudie durchgeführt, bei der der Einfluss der Modellpriorisierung auf die „time to report“, also die Dauer bis zur nächsten Aktion, analysiert wurde.

In der Simulation wurde ein Thorax-Röntgenbild nach dessen Aufnahme durch das Modell priorisiert und dementsprechend in eine Warteschlange eingeordnet.

Die mittlere time to report konnte für kritisch priorisierte Bilder von knapp 11 auf rund 3 Tage reduziert werden.

Die Autoren fügten zusätzlich falsch klassifizierte Bilder in die Warteschlange ein, um realitätsnähen Noise zu simulieren. Selbst dann wurde noch eine Verbesserung bei kritischen und dringenden Fällen festgestellt. Innerhalb der ersten 24 Stunden wurden so für 85% dieser Fälle weitere Schritte eingeleitet, während dies bei den historischen Daten nur für 60% erfolgte. Wie zu erwarten – und auch erwünscht – war, verlängerte sich hingegen die time to report für Aufnahmen normaler Priorität.

Die Fehlklassifikationen wurden im Nachgang erneut untersucht. Dabei wurde festgestellt, dass von den fünf kritischen Fällen, die als normal klassifiziert wurden, das Modell bei vier Bildern richtig lag und bei den 95 dringenden aber als normal klassifizierten Fällen, die Modellvorhersage bei 36 Fällen korrekt war. Der tatsächliche Fehler entstand demnach bei der Labelerzeugung.

Die Autoren weisen darauf hin, dass selbst die KI-gestützte Priorisierung mit einer time to report von 2.7 Tagen für kritische und 4.1 Tagen für dringende Fälle für Nord-Amerikanische Verhältnisse inakzeptabel ist .

### 7.2.5 Limitationen

Annarumma et al. [11] führen mehrere Limitationen auf, die im Folgenden kurz dargestellt werden.

Zunächst besteht weiterhin ein (geringes) Risiko, dass Fälle falsch priorisiert werden. Je nachdem wie weit fortgeschritten eine Pathologie ist, kann sich die Priorität auch ändern. Dies wird aber nicht durch die Label abgebildet. Außerdem könnte das Modell die richtige Priorität vorhersagen, obwohl es die falsche Pathologie erkannt hat, also aus den falschen Gründen die richtige Entscheidung treffen. Die Bilder werden darüber hinaus ohne Kontext, wie beispielsweise bereits stattfindende (ambulante) Behandlung, betrachtet. Dies kann die tatsächliche Priorität ändern. Zudem schätzen die Autoren, dass 3-5% der Label fehlerhaft sind, da die radiologischen Untersuchungen nicht trivial und nicht eindeutig sind und ferner enthält der Datensatz keine Röntgenaufnahmen von bereits stationierten Patienten.

## 7.3 Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels

Pham et al. [95] stellen eine Methodik zum Modelltraining für die Klassifikation von Thorax-Röntgenbildern vor, die versucht, die hierarchischen Beziehungen zwischen den einzelnen Pathologien zu berücksichtigen.

Sie verwenden für ihre Studie den CheXpert-Datensatz. Dieser beinhaltet auch „uncertain“ Labels (vgl. Abschnitt 5.4). Irvin et al. [17] betrachten diese entweder als negativ („U-Zeros“) bzw. positiv („U-Ones“), ignorieren sie („U-Ignore“), nutzen sie als eigene Klasse oder lassen sie durch das Modell selbst labeln, nachdem ein U-Ignore angewendet wurde („U-SelfTrained“). Dabei werden allerdings entweder mit großer Wahrscheinlichkeit falsche Label erzeugt, oder Daten komplett ignoriert. Stattdessen nutzen Pham et al. [13] eine Label-Smoothing-Regularisierung (LSR) um die Ungewissheit besser auszudrücken.

### 7.3.1 Label-Smoothing-Regularisierung (LSR)

Für die LSR werden die uncertain Labels nicht auf einen festen Wert gesetzt, sondern die neuen Label zufällig aus einem gewissen Wertebereich gleichverteilt gezogen. Um die U-Ones-Strategie zu glätten (engl. *smoothen*) wird dafür der Wertebereich  $U(a_1, b_1)$  über die Hyperparameter  $a_1$  und  $b_1$  festgelegt und analog für U-Zeros  $U(a_0, b_0)$ . So ergeben sich die Ansätze U-Ones+LSR und U-Zeros+LSR. Aus empirischen Gründen wählten die Autoren  $a_1 = 0.55$ ,  $b_1 = 0.85$ ,  $a_0 = 0$  und  $b_0 = 0.3$ . Die LSR verhindert dadurch auch, dass sich das Modell auf den uncertain Daten zu sicher ist (engl. *overconfident*), da die Zielvariable nicht mehr eindeutig positiv bzw. negativ ist.

### 7.3.2 Conditional Training

Zwischen den einzelnen Labeln des CheXpert-Datensatzes existiert eine gewisse Hierarchie, die mit ausreichend domänen spezifischem bzw. medizinischem Fachwissen als Baum oder gerichteter Graph dargestellt werden kann. Die Autoren nutzen die Labelbeziehungen von Irvin et al. [17] um einen Graphen zu erstellen, der in Abbildung 7.2 gezeigt wird. Damit umfassen die Eltern-Knoten die Label Enlarged Cardiomegaly sowie Lung Opacity und Consolidation.

Das Modell-Training wird bei dieser Methode in zwei Schritte aufgeteilt. Im ersten, dem *Conditional Training*, wird das Modell nur auf den Daten trainiert, bei denen die Eltern-Label positiv sind. Wenngleich es aus der Veröffentlichung nicht klar so hervorgeht, ergab eine Nachfrage bei den Autoren, dass in diesem Fall die Daten so gefiltert werden, dass nur Datenpunkte vorkommen, bei denen sowohl Lung Opacity als auch Consolidation positiv sind. Dafür wird eine logische AND-Verknüpfung eingesetzt. Ziel dieser Phase ist es, dass das Modell lernen soll, zwischen den Kind-Knoten zu unterscheiden.

Anschließend wird Transfer Learning verwendet, und das vortrainierte CNN auf dem gesamten Datensatz trainiert. Dabei soll das Netz lernen, die Eltern-Label zu erkennen. Dafür frieren die Autoren die Parameter aller Schichten außer der Ausgabe-FC-Schicht

ein. Sie nehmen demnach ein Fine-tuning vor.

Die Modell-Outputs sollten danach als die *bedingte* Wahrscheinlichkeit betrachtet werden, dass ein Label positiv ist, unter der Bedingung dass seine Eltern positiv sind. Für die Praxis möchte man aber unbedingte Wahrscheinlichkeiten, was sich durch die Anwendung der Bayes'schen Regel [32, S. 68],

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)}, \quad (7.2)$$

erreichen lässt. Zur Veranschaulichung sei  $A$  das Vorkommen von Pneumonia,  $B$  von Consolidation und  $C$  von Lung Opacity. Betrachten wir als Beispiel die unbedingte Wahrscheinlichkeit für das Label Pneumonia  $P(A)$ . Die beiden Eltern Consolidation und Lung Opacity sollten in diesem Fall beide positiv sein (vgl. Abbildung 7.2). Demnach ist die Wahrscheinlichkeit  $P(A) = P(A, B, C)$ .

Das Modell liefert uns  $P(A|B, C)$  als Ausgabe für das Label Pneumonia und analog  $P(B|C)$  für Consolidation, sowie  $P(C)$  als Ausgabe für das Label Lung Opacity. Außerdem ist  $P(B|A) = 1$ , da  $B$  ein Eltern-Knoten von  $A$  ist und mit der gleichen Begründung gilt  $P(C|B) = 1$ . Somit berechnen wir mittels Gleichung (7.2) die Wahrscheinlichkeit, dass Pneumonia positiv ist unter der Bedingung, dass Consolidation positiv ist als

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} = \frac{P(A)}{P(B)}, \quad (7.3)$$

und somit gilt

$$P(A) = P(A|B) \times P(B). \quad (7.4)$$

Außerdem lässt sich die Wahrscheinlichkeit von Consolidation unter der Bedingung, dass Lung Opacity vorliegt, berechnen als

$$P(B|C) = \frac{P(B)P(C|B)}{P(C)} = \frac{P(B)}{P(C)}, \quad (7.5)$$

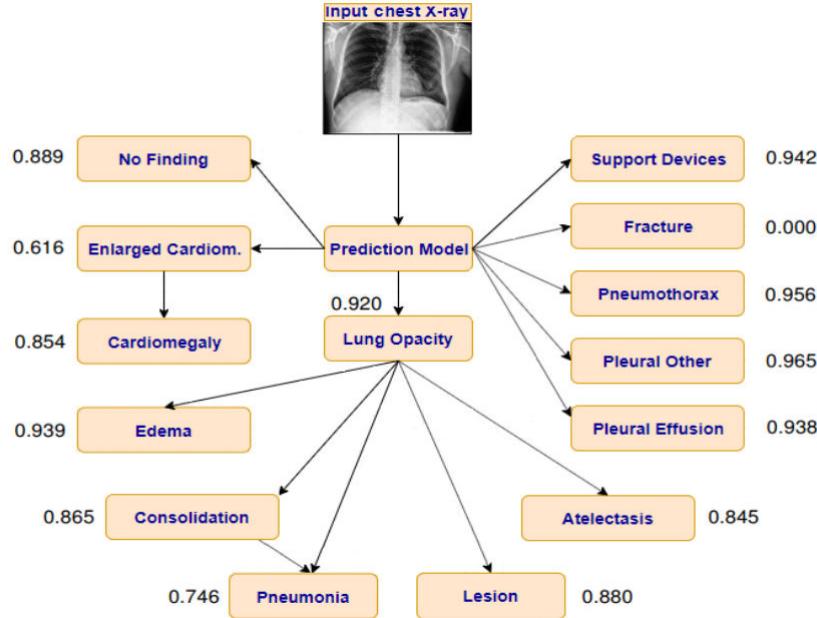
wobei

$$P(B) = P(B|C) \times P(C). \quad (7.6)$$

Aus den Berechnungen (7.4) und (7.6) ergibt sich schließlich die unbedingte Wahrscheinlichkeit, dass das Label Pneumonia positiv ist als

$$P(A) = P(A|B) \times P(B|C) \times P(C). \quad (7.7)$$

Das gleiche Prinzip kann auf beliebig lange Eltern-Kind-Ketten angewendet werden. Dieses Vorgehen führt zudem dazu, dass die Wahrscheinlichkeit eines Kind-Knotens immer maximal so groß ist wie die Wahrscheinlichkeit des Eltern-Knotens, was der medizinischen Realität entspricht. Hier ergab sich aus der Nachfrage bei den Autoren, dass die Anwendung der Bayes'schen Regel allerdings nicht zwingend zu besseren Resultaten führte.



**Abbildung 7.2:** Hierarchie zwischen den Labels des CheXpert-Datensatzes. Quelle: [95].  
Die Pfeile zeigen von Eltern- zu Kindknoten

### 7.3.3 Datenvorbereitung

Pham et al. [95] erproben einen Template Matching Algorithmus, um den Bildbereich auszuschneiden, der die Lunge enthält. Zunächst werden die Bilder auf die Auflösung  $256 \times 256$  skaliert und anschließend mittels Template Matching ein Bereich von  $224 \times 224$  Pixeln ausgeschnitten. Zudem wurden die Bilder mit Mittelwert und Standardabweichung des ImageNet-Datensatzes normalisiert, da die Autoren die Gewichte von einem auf diesen Daten vortrainierten Modell nutzen.

### 7.3.4 Modelltraining

Die Autoren verglichen Ihre U-Zeros+LSR und U-Ones+LSR mit den Methoden U-Ignore, U-Zeros und U-Ones. Nachdem jeweils eine dieser Herangehensweisen genutzt wurde, um die uncertain Label zu behandeln, wurde mit einem auf ImageNet vortrainierten DenseNet-121 das Conditional Training durchgeführt. Die Autoren nutzen nach der letzten FC-Schicht wie in Abschnitt 2.6.1 beschrieben die Sigmoid-Funktion. Als Optimisierer kam Adam [96] mit Standardparametern zum Einsatz, eine Lernrate von  $1e-4$ , die nach jeder Epoche um den Faktor 10 reduziert wurde, eine Batchgröße von 32 und der Binary-Cross-Entropy (BCE) Loss als Verlustfunktion. Zunächst wurde das Conditional Training für fünf Epochen ausgeführt, dann alle Schichten bis auf die Output-Schicht eingefroren und für fünf Epochen auf allen Trainingsdaten trainiert [95]. Zur Bewertung der Güte wird der (ROC-)AUC-Score genutzt (vgl. Abschnitt 2.6).

### 7.3.5 Nutzung von Ensembling-Methoden

Die Autoren argumentieren, dass die AUC eines Modells stark von dessen Architektur abhängt und bilden daher ein Ensemble bestehend aus mehreren Architekturen. Sie nutzen ein DenseNet-121, DenseNet-169 und DenseNet-201, die von Huang et al. [30] vorgestellt wurden, ein Inception-ResNet-v2 von Szegedy et al. [97], ein Xception-Netz von Chollet et al. [98] sowie ein NASNetLarge von Zoph et al. [99]. Insgesamt werden demnach sechs Modelle trainiert, deren Vorhersagen die Autoren mittelten, um die Ensembling-Vorhersage zu erhalten. Es wurde also ein Soft-Voting-Ensemble (vgl. Abschnitt 2.5) eingesetzt.

Eine Besonderheit gilt es bei dem Inception-ResNet-v2 zu berücksichtigen. Die Autoren beschreiben, dass sie eine Bildauflösung von  $224 \times 224$  verwenden. Damit ist es allerdings nicht möglich, ein Inception-ResNet-v2 zu trainieren, da dieses die Inputs dafür während des Forward-Passes zu stark verkleinert. Das führt dazu, dass sich innerhalb des Netzwerks Schichtenausgaben der Größe  $0 \times 0$  ergeben, was ein Fortführen des Forward-Passes und somit das Training verhindert. Das Inception-ResNet-v2 erwartet Eingaben mit einer Auflösung von mindestens  $299 \times 299$ . Unter der wahrscheinlichen Annahme, dass die Autoren dieses Modell entsprechend trainiert haben, lässt sich feststellen, dass sie in ihrem Ensemble ähnlich zu Annarumma et al. [11] zusätzlich zu diversen Architekturen auch unterschiedliche Bildauflösungen verwenden.

Des Weiteren nutzen die Autoren bei der Inferenz eine Test-Time-Augmentation (TTA). Dabei wenden sie zehn mal zufällige Bildaugmentierungen an und bilden den Mittelwert der zehn einzelnen Vorhersagen um die endgültige Ausgabe zu erhalten. Zu den Augmentierungen zur Testzeit gehörten dabei horizontales Spiegeln, Rotieren ( $\pm 7$  Grad), Scheren ( $\pm 5$  Pixel) und Skalieren ( $\pm 2\%$ ).

### 7.3.6 Resultate

Pham et al. erreichten mit ihrem Ansatz auf dem geheimen Testset der CheXpert Competition einen mittleren AUC-Wert von 0.93 auf den fünf Vergleichslabeln Atelectasis, Cardiomegaly, Consolidation, Edema, und Pleural Effusion. Ihr Modell war besser als 2.6 der 3 Vergleichsradiologen und belegte zeitweise den ersten Platz bzw. zum Zeitpunkt der Entstehung dieser Arbeit den zweiten Platz der CheXpert Competition. Sie erreichen zwar den gleichen AUC-Wert wie der Erstplatzierte, allerdings übertrifft dieser durchschnittlich mehr Radiologen und belegt daher den ersten Platz. Auf den Validierungsdaten erreichte das beste einzelne Modell einen mittleren AUC von 0.894 und das Ensembling führte mit einem AUC-Wert von 0.940 zu deutlichen Verbesserungen.

Die Autoren führten außerdem Ablationsstudien durch, um die Effekte der einzelnen Schritte zu beurteilen.

Dabei stellte sich heraus, dass das Template Matching und die TTA nur zu marginalen Verbesserungen des AUC-Wertes führten. Dies bekräftigt empirisch den positiven Effekt des Conditonal Training und der LSR.

Des Weiteren weisen Pham et al. [95] darauf hin, dass es ohne Ensembling unmöglich scheint, eine derartig gute Performance zu erreichen.

### 7.3.7 Limitationen

Die von Pham et al. [95] aufgeführten Limitationen werden im Folgenden kurz aufgezeigt.

Zunächst ist zu berücksichtigen, dass die Bilder aus dem verwendeten Datensatz stammen nur von einem Krankenhaus. Die Modelle generalisieren vermutlich nicht genauso gut auf Datensätzen anderer Kliniken mit anderen Röntgengeräten, was als geographische Variation bezeichnet wird.

Außerdem begutachten Radiologen für die Diagnose in der Realität eine Vielzahl an Informationsquellen, wie Alter und medizinische Vergangenheit, und nicht nur ein einziges Röntgenbild. Dies wird vom Modell ignoriert.

Die Autoren schlagen vor, auch Modelle mit höheren Bildauflösungen zu trainieren, was aber zu stark erhöhtem Rechenaufwand führt.

Der Datensatz beinhaltet zudem eine größere Menge Bilder von sehr geringer Qualität, die das Lernen beeinträchtigen können. Das Template Matching ist nicht ausreichend mächtig, um diese zu entfernen und sollte daher durch eine OoD-Erkennung ersetzt werden.

## 7.4 Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification

Guan et al. [100] stellen eine Methodik vor, die durch mehrere Zweige (engl. *Branches*) gleichzeitig das gesamte Bild sowie den wichtigsten Bildausschnitt berücksichtigen soll. Dies bezeichnen sie als Attention Guided Convolutional Neural Network (AG-CNN).

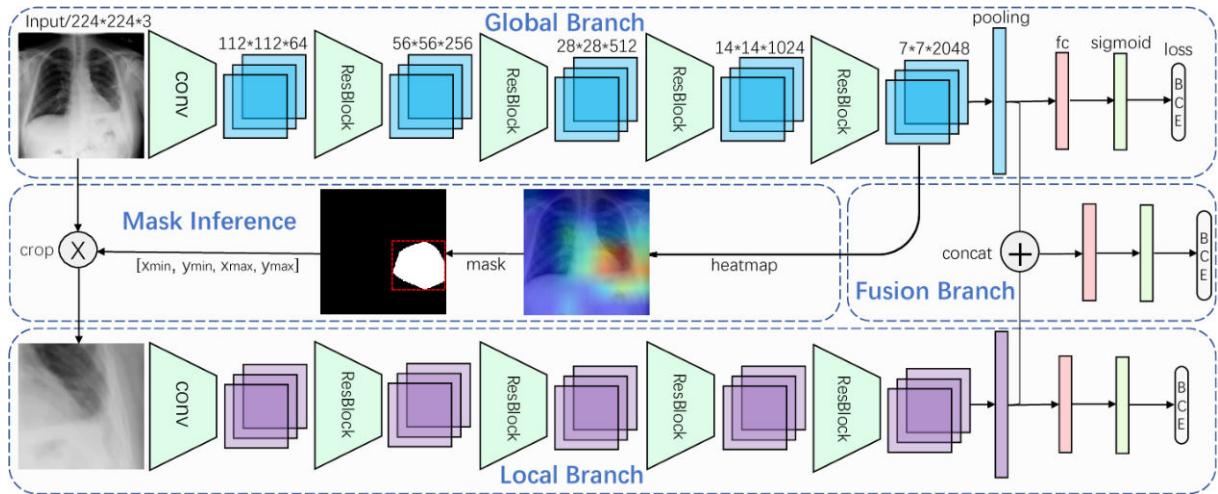
### 7.4.1 Architektur

Die Architektur besteht aus drei Branches: Dem *Global Branch (GB)*, *Local Branch (LB)* und *Fusion Branch (FB)* [100].

Der globale und lokale Branch sind dabei identisch aufgebaut und bestehen aus entweder einem ResNet-50 oder einem DenseNet-121, die als *Backbone* bezeichnet werden. Sie unterscheiden sich lediglich hinsichtlich ihrer Eingaben. Während der globale Branch das gesamte Röntgenbild erhält, wird für den Local Branch ein lokaler Bildausschnitt erzeugt. Diese beiden Betrachtungsweisen sollen sich ergänzen. Global und Local Branch werden mit den vortrainierten Gewichten von ImageNet initialisiert und im Training Fine-tuned.

Der Fusion Branch soll dann, wie der Name schon sagt, die beiden anderen Branches kombinieren. Er besteht aus einer einzelnen FC-Schicht und bekommt als Input die Ausgaben der letzten Pooling-Schichten des globalen und lokalen Branches und erzeugt den Output des AG-CNN.

Alle drei Branches verwenden als Klassifizierungsschicht eine FC-Schicht mit einem Ausgabeneuron für jedes mögliche Label, auf das die Sigmoid-Aktivierungsfunktion angewendet wird, und werden mit dem BCE Loss trainiert.



**Abbildung 7.3:** Architektur des Attention-Guided CNN mit Global, Local und Fusion Branch sowie der Erzeugung von Binärmasken aus den Aktivierungen des Globalen Branches. Quelle: Guan et al. [100].

Die Autoren bilden hier also ein Ensemble, das dem Stacking zugeordnet werden kann, da mit dem Fusion Branch ein eigenes Modell trainiert wird, das die Ausgaben von Global und Local Branch kombiniert und die Ensembling-Vorhersage erzeugt. Außerdem sind in dem Konzept die Läsionsbereiche für die Verbesserung der Klassifikation zu nutzen, Parallelen zu dem Ansatz von Ye et al. [76] festzustellen (vgl. Abschnitt 6.1.3).

Die gesamte Architektur wird in Abbildung 7.3 dargestellt.

#### 7.4.2 Erzeugen lokaler Bildausschnitte

Guan et al. [100] erzeugen die lokalen Bildausschnitte, die als Eingabe für den Local Branch dienen, werden aus den Aktivierungen des globalen Branches. Dafür wird zunächst aus den Logits der letzten Convolutional-Schicht eine Heatmap  $H_g$  erzeugt, indem das Maximum entlang der Kanäle gebildet wird und diese anschließend auf die Bildgröße skaliert werden. Wird beispielsweise ein ResNet-50 als Architektur verwendet, hat der Output der letzten Convolutional-Schicht eine Dimension von  $7 \times 7 \times 2048$  (vgl. Abbildung 7.3). Durch das Selektieren des Maximums ergibt sich eine Dimension von  $7 \times 7$ , die dann mittels Interpolation auf  $224 \times 224$  vergrößert wird.

Die Heatmap kann also als eine Form von Aufmerksamkeit (engl. *Attention*) aufgefasst werden, da die Bereiche höherer Aktivierung wichtiger für die Klassifizierung sind. Daraufhin wird aus der Heatmap eine Binärmaske  $M$  als *Region-of-Interest* (ROI) erzeugt, indem die Werte der Heatmap  $H_g(x, y)$  an den Positionen  $(x, y)$  mittels eines Schwellwerts  $\tau$  auf 1 bzw. 0 abgebildet werden. Die Binärmaske ergibt sich also über

$$M(x, y) = \begin{cases} 1, & \text{wenn } H_g(x, y) > \tau, \\ 0, & \text{sonst.} \end{cases} \quad (7.8)$$

Offensichtlich spielt der Schwellwert  $\tau$  dabei eine entscheidende Rolle. Je größer dieser ist, desto kleiner wird die ROI. Je kleiner er ist, desto ähnlicher ist die Eingabe für den

Branch	Mittlere AUC ResNet-50	Mittlere AUC DenseNet-121
Global Branch	0.841	0.840
Local Branch	0.817	0.810
Fusion Branch	0.868	0.871

**Tabelle 7.1:** Vergleich der AG-CNN Branches für ResNet-50 und DenseNet-121 Backbones aus [100].

lokalen Branch zu dem globalen Bild.

Von der Binärmaske wird dann die maximale verbundene Fläche (engl. *maximum connected region*) verwendet, um das Rechteck  $[x_{min}, y_{min}, x_{max}, y_{max}]$  zu definieren, dass alle relevanten Punkte enthält.

Da die Maske binär ist, kann durch die Multiplikation von  $M$  mit dem zugehörigen Eingabebild ein Bildausschnitt erzeugt werden. Mit diesem Bildausschnitt, der ausschließlich den Bereich enthält, der am wichtigsten für den globalen Branch war, wird im folgenden Schritt der Local Branch trainiert.

Das Ausschneiden der ROI erfolgt demnach unüberwacht. Der von den Autoren verwendete Datensatz enthält für einzelne Bilder Informationen über Bounding-Boxes, die die korrekte ROI definieren. Einen von Guan et al. [100] gewählten Vergleich zwischen diesen vorgegebenen und den aus den Heatmaps erzeugten ROIs befindet sich in Abbildung 7.7. Dieser zeigt, dass die Methodik zwar nicht perfekt, aber doch in beachtlicher Genauigkeit funktioniert.

### 7.4.3 Durchgeführte Experimente

Guan et al. [100] führten zur Beurteilung ihrer Methode mehrere Experimente durch. Alle Experimente wurden dabei auf dem ChestX-ray14-Datensatz vollzogen und eine Auswahl wird im Folgenden beschrieben.

#### Vergleich der Branches und Backbone-Architektur

Der wichtigste Vergleich ist der der einzelnen Branches. In Tabelle 7.1 werden die durchschnittlichen AUC-Werte der Branches für beide Backbone-Architekturen dargestellt. Dabei zeigt sich unabhängig des Backbones, dass der lokale Branch zwar schlechter ist als der globale Branch, der Fusion Branch jedoch immer zu den besten Ergebnissen führt. Die insgesamt beste AUC wird von dem Fusion Branch des DenseNets-121 erreicht, wobei die Unterschiede zwischen den Backbones gering sind.

#### Trainingsreihenfolge der Branches

Ein weiteres Experiment betraf die Trainingsreihenfolge der Branches. Die Autoren verglichen dabei fünf verschiedene Strategien mit einem ResNet-50 als Backbone.

*GL\_F* beschreibt, dass zunächst der globale und lokale Branch gemeinsam trainiert werden, dann der Fusion Branch. Dagegen werden bei *GLF* alle drei Branches zusam-

Strategy	Batchsize	Global	Local	Fusion
GL_F	64/64/64	0.831	0.800	0.833
GLF	64/64/64	0.847	0.815	0.849
G_LF	128/64/64	0.841	0.809	0.843
G_L_F*	128/64/64	0.852	0.819	0.854
G_L_F	128/64/64	0.841	0.817	0.868

Abbildung 7.4: Resultat des Vergleichs der Trainingsstrategie für das AG-CNN. Quelle [100].

Statistic	Global	Local	Fusion
Max	0.8412	0.8171	0.8680
L1	0.8412	0.8210	0.8681
L2	0.8412	0.8213	0.8672

Abbildung 7.5: Resultat des Vergleichs von Guan et al. [100] zur Strategie zur Erzeugung der Heatmaps aus den Aktivierungen.

men trainiert. Analog zur ersten Strategie wird bei *G\_LF* zuerst der globale Branch trainiert, dann die anderen beiden gemeinsam. Bei *G\_L\_F* und *G\_L\_F\** werden die drei Branches sequenziell trainiert. Der Asterisk bedeutet, dass Global und Local Branch während dem Trainieren des Fusion Branches Fine-tuned werden.

Die Resultate dieses Experiments befinden sich in Abbildung 7.4.

Es lässt sich erkennen, dass die Performanz des Fusion Branches für die Strategie *G\_L\_F* am besten ist und diese auch die insgesamt höchsten AUC-Werte erzielt. Der globale und lokale Branch sind für *G\_L\_F\** am besten und verlieren bei *GL\_F* und *GLF* am stärksten an Genauigkeit.

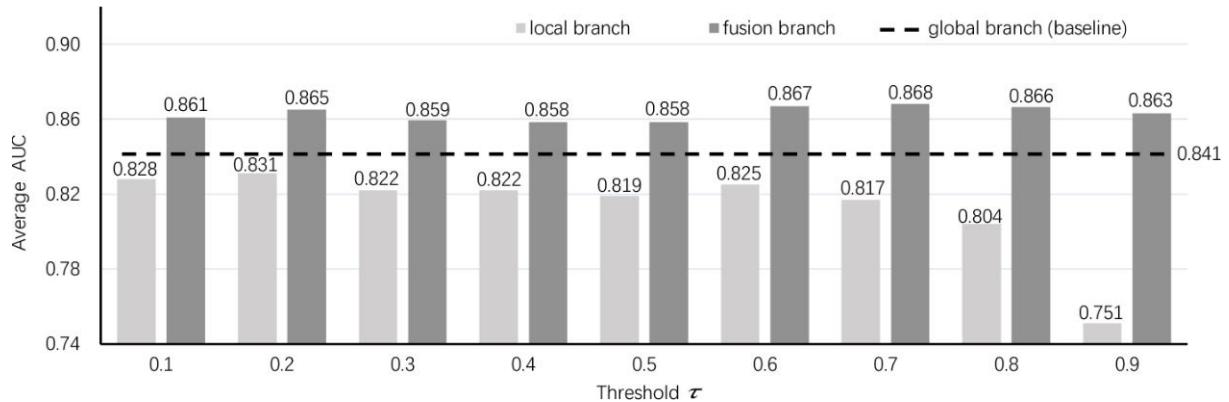
### Strategie zur Heatmap-Erzeugung

Neben der Nutzung des Maximums wurden von den Autoren auch die L1- und L2-Norm mit einem ResNet-50 als Backbone erprobt. Die Resultate befinden sich in Abbildung 7.5.

Dabei stellte sich heraus, dass die Unterschiede sowohl auf dem Local als auch auf dem Fusion Branch marginal sind. Da die Maximumsbildung das einfachste Vorgehen ist, entschieden sich die Autoren daher für diesen Ansatz.

### Auswirkung des Schwellwertes zur Binärmasken-Erzeugung

Der Schwellwert  $\tau$  wirkt sich direkt auf die Größe der ROI und somit auf den Input für den Local Branch aus. Ist  $\tau$  sehr klein, wird der Bildausschnitt sehr groß und der lokale Branch ähnelt demnach stärker dem globalen Branch.



**Abbildung 7.6:** Vergleich der gemittelten AUC-Werte für den Local und Fusion Branch für verschiedene Schwellwerte  $\tau$  auf den Testdaten aus den Experimenten von Guan et al. [100]. Der Global Branch kann als Baseline betrachtet werden, da er von den Änderungen unberührt bleibt.

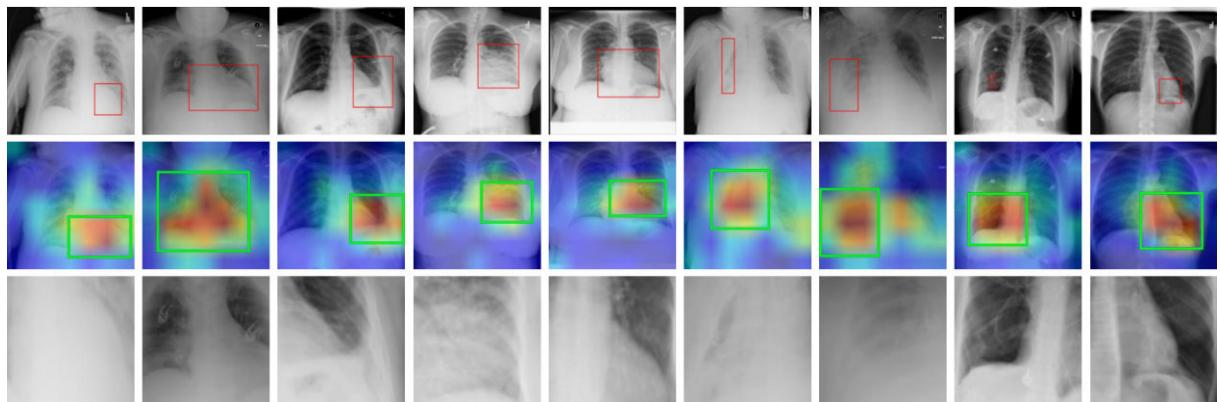
Die Autoren haben für verschiedene  $\tau$  die durchschnittlichen AUCs des Local und Fusion Branches verglichen, das Ergebnis befindet sich in Abbildung 7.6. Es zeigt, dass die AUCs des lokalen Branches für sehr kleine  $\tau$  am nächsten an denen des globalen Branches liegen. Die AUCs des Fusion Branches erreichen ihre höchsten Werte dagegen für ein  $\tau$  von 0.6 – 0.8. Dies deutet darauf hin, dass der Local Branch am meisten beiträgt, wenn er nur einen kleineren Bildbereich verwendet. Das Experiment zeigt darüber hinaus, dass der Fusion Branch nicht so stark beeinflusst wird, wie der Local Branch, und eine gewisse Resistenz gegenüber Hyperparameteränderungen aufweist.

#### 7.4.4 Resultate

Guan et al. [100] stellten mit dem mittleren AUC von 0.871 zum Zeitpunkt der Veröffentlichung einen neuen Stand der Technik für den ChestX-ray14-Datensatz auf. Für 13 der 14 Label erzielten sie einen besseren AUC-Wert als die bisherigen Veröffentlichungen. Außerdem zeigten die Autoren, dass ihr Ansatz unabhängig von der Backbone-Architektur funktioniert und robust gegenüber Hyperparameteränderungen ist.

#### 7.4.5 Nutzung von Ensembling-Methoden

Durch den Fusion-Branch bilden die Autoren ein Ensemble aus dem globalen und dem lokalen Branch. Somit kann dies dem Stacking zugeordnet werden, da ein eigenes Modell trainiert wird, das selbst lernen soll, wie die Ensemble-Mitglieder zu kombinieren sind. Eine Besonderheit ist dabei die Abhängigkeit der Trainingsdaten des LB vom GB und somit zwischen den Ensemble-Mitgliedern. In der Eigenschaft, dass der LB auf Grundlage des GB trainiert wird, lässt sich eine Ähnlichkeit zu Boosting-Methoden erkennen.



**Abbildung 7.7:** Vergleich zwischen den gelabelten Bounding-Boxes (erste Zeile, rot) und den aus den Aktivierungen erzeugten Heatmaps mit den ROIs (zweite Zeile, grün). Die dritte Zeile zeigt den ausgeschnittenen und skalierten Bildbereich. Quelle: Guan et al. [100].

## 7.5 A Novel Approach for Multi-Label Chest X-Ray Classification of Common Thorax Diseases

Allaouzi et al. [52] verwenden ein CNN als Feature Extractor mit einer Logistischen Regression als Klassifizierer und stellen einen Vergleich zwischen den verschiedenen Methoden zur Problemtransformation an.

Sie verwenden für ihre Experimente den ChestX-ray14-Datensatz, sowie den CheXpert-Datensatz. Da die Autoren aber Bilder mit uncertain Labels verwerfen, reduziert sich der CheXpert-Datensatz auf etwas mehr als die Hälfte.

Die Autoren skalieren die Bilder auf eine Auflösung von  $224 \times 224$  Pixel und wenden zufällig horizontales Spiegeln sowie eine Normalisierung mit Mittelwert und Standardabweichung an. Die Autoren spezifizieren nicht explizit, ob sie Mittelwert und Standardabweichung des jeweiligen Thorax-Röntgenbilder-Datensatzes verwenden, oder die des ImageNet-Datensatzes. Da sie aber ein CNN, das auf ImageNet vorgenommen wurde, Fine-tunen, wird vermutlich mit den Werten von ImageNet normalisiert. Als Backbone verwenden die Autoren ein vortrainiertes DenseNet-121. Dieses trainieren sie mit einer Batch-Größe von 8 für 110 Epochen unter Verwendung des Binary-Cross-Entropy-Losses und des Adam Optimisierers mit einer initialen Lernrate von 0.001, die mit 10 multipliziert wird, nachdem der Validierungsloss ein Plateau erreicht hat. Außerdem teilen sie die Datensätze in je 80% für das Training und 20% als Testdatensatz auf. Dieser Ansatz verwendet mit der hohen Epochenzahl, geringen Batch-Größe und steigenden Lernrate eine im Vergleich zu den übrigen Methoden ungewöhnliche Trainingskonfiguration. Die Batch-Größe dürfte sich teilweise durch die verwendete Hardware erklären lassen, da die von den Autoren genutzte Grafikkarte nur 4 Gigabyte Speicher zur Verfügung hat.

Nachdem dieses Fine-tuning abgeschlossen ist, wird die letzte Schicht des DenseNets entfernt. Dieses gibt nun einen Feature-Vektor der Dimension 1024 zurück, der durch das AveragePooling entstanden ist. Dieser wird als Eingabe für eine Logistische Regression verwendet, die mittels Binary Relevance, Classifier Chain und Label Powerset

(vgl. 3.2) in die Lage versetzt wird, eine Multi-Label Klassifikation durchzuführen.

### 7.5.1 Resultate

Allaouzi et al. [52] erreichten auf dem ChestX-ray14-Datensatz mit allen drei Problemtransformations-Methoden einen mittleren AUC von gerundet 0.88. Auf dem CheXpert-Datensatz erreichen sie eine mittleren AUC über alle Label von 0.812, 0.808 und 0.785 für BR, LP und CC [52]. Für die fünf Vergleichslabel des CheXpert-Datensatzes erreicht die Methode mit BR einen durchschnittlichen AUC von 0.828. Allerdings ist hierbei zu berücksichtigen, dass die Autoren diese Werte auf ihrem eigens definierten Testdatensatz und nicht auf dem designierten Validierungs- oder geheimen Testset erreicht haben, was die Vergleichbarkeit zu anderen Veröffentlichungen erheblich einschränkt. Zudem haben die Autoren nur den Teil des Datensatzes verwendet, der keine uncertain Label enthält.

Dennoch kann diese Veröffentlichung für die weiteren Schritte der vorliegenden Arbeit interessant sein. Denn wie in Abschnitt 2.5 erläutert, sollten die Mitglieder eines Ensembles möglichst divers sein und unterschiedliche Fehler erzeugen, um den Gesamtfehler zu reduzieren. Es ist zu erwarten, dass die Logistische Regression andersartige Fehler erzeugt als die CNNs, wodurch ihre Nutzung als Ergänzung des Ensembles sinnvoll sein könnte.

### 7.5.2 Nutzung von Ensembling-Methoden

Die Autoren nutzen keine Ensembling-Methoden. Dies würde sich besonders auf Seite der Klassifizierer anbieten, da neben einer Logistischen Regression auch andere Modelle wie beispielsweise eine Support-Vector-Machine (SVM) oder Random-Forests, denen das Ensembling inhärent ist, genutzt werden könnten. Auch auf Seite der Feature Extraction wäre es denkbar, mehrere Modelle zu erzeugen und mit deren Ausgaben einen oder mehrere Klassifizierer zu trainieren.

## 7.6 Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs

Kumar et al. [12] legen ihren Fokus auf die Kookkurrenzen zwischen den Pathologien des ChestX-ray14-Datensatzes (vgl. Abbildung 5.2) und der Verbesserung der Klassifizierung durch Boosting. Bezuglich der Kookkurrenzen vergleichen die Autoren die Problemtransformation BR und den *PariWise Error (PWE)* Loss. Das Boosting führen sie durch eine Kaskadierung von CNN und FC-Schichten durch. Außerdem nutzten sie Over- und Undersampling um die Klassen-Imbalance auszugleichen.

### 7.6.1 Binary Relevance und PWE Loss

Kumar et al. [12] argumentieren, dass die Kookkurrenzen der Pathologien Information enthält, die die Klassifizierung verbessern kann. Die Binary Relevance Problemtrans-

formation (vgl. Abschnitt 3.2), die häufig genutzt wird um Multi-Label Klassifikation zu ermöglichen, berücksichtigt dies jedoch nicht. Daher vergleichen sie diesen Ansatz mit dem des Training durch den smooth PWE Loss aus von Li et al. [101], durch dessen paarweisen Vergleich zwischen jeder positiven und negativen Klasse das Modell die Beziehungen der Klassen lernen soll. Um die Klassen-Imbalance auszugleichen, gewichteten sie den Cross Entropy Loss und führten für die Mehrheitsklasse Under- und die Minderheitsklasse Oversampling durch.

Mit dem Input  $x$  und den positiven Labeln  $Y$ , der Modellfunktion  $f(x)$  und  $f_u(x)$  als das  $u$ -te Element von  $f(x)$  lässt sich der smooth PWE Loss über

$$\text{PWE} = \log \left( 1 + \sum_{v \notin Y_i} \sum_{u \in Y_i} \exp(f_v(x_i) - f_u(x_i)) \right) \quad (7.9)$$

berechnen, wobei mit  $[.]_i$  über die Datenpunkte iteriert wird. Zentral bei der Berechnung ist also die Differenz zwischen der Modellvorhersage für die negative Klasse  $f_v(x_i)$  und positiven Klasse  $f_u(x_i)$ . Die Formulierung dieser Verlustfunktion führt nach Li et al.[101] dazu, dass die Funktion  $f(x)$  die Eigenschaft erfüllt, dass ihre Ausgaben für positive Label größer sind als für negative:

$$f_u(x_i) > f_v(x_i), \quad \forall u \in Y, v \notin Y. \quad (7.10)$$

## 7.6.2 Boosted cascaded convnets

Für beide Ansätze (BR und PWE) implementierten Kumar et al. [12] zusätzlich eine *Boosted Cascade*. Das Boosting ist ein iteratives Verfahren, bei dem ein Modell in jeder Iteration verbessert werden soll (vgl. Kapitel 2.5.3).

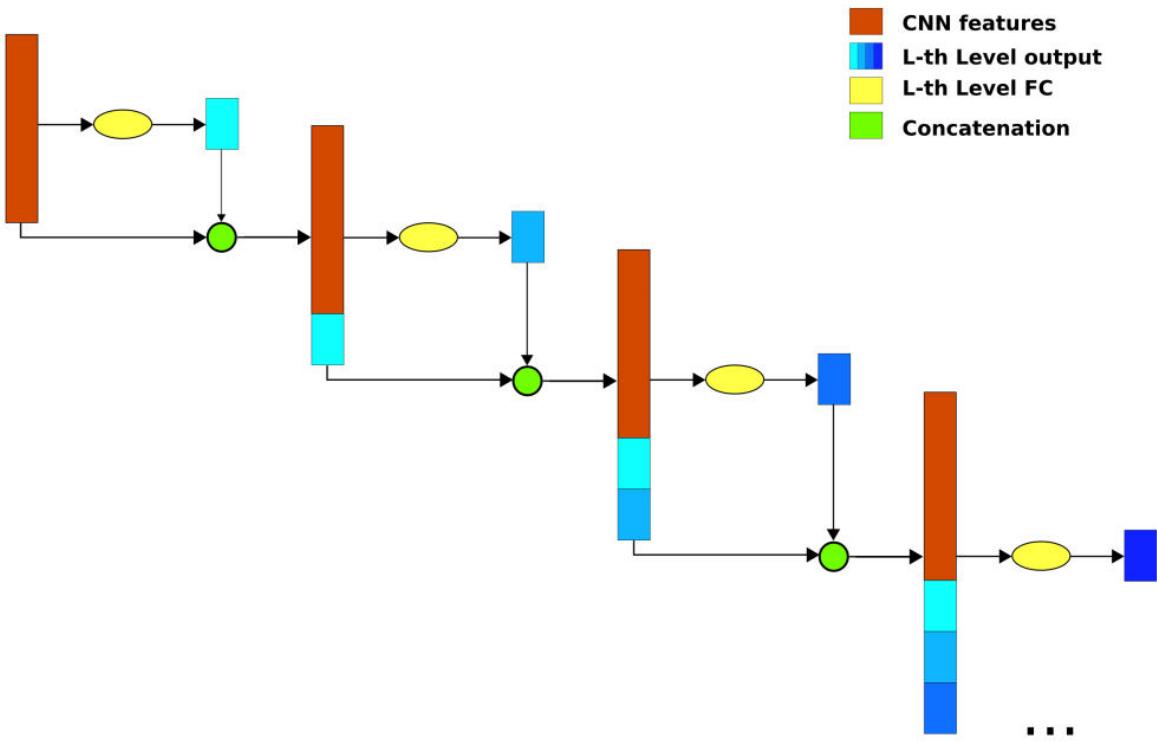
Die Autoren implementieren dies, indem jeder Klassifizierer die Vorhersagen aller vorheriger Modelle erhält. Dabei wird jedes Modell auf dem vollen Trainings-Datensatz trainiert. Beim Boosting ist es außerdem typisch, dass jede Iteration ein stärkeres Gewicht auf die schwierigen, also von dem vorherigen Modell „schlechter“ klassifizierten, Daten legen soll. Dies setzten die Autoren um, in dem sie ein gewichtetes Sampling anhand der Schwierigkeit durchführen.

Die Wahrscheinlichkeit  $p_i$ , dass der  $i$ -te Datenpunkt aus einem Datensatz mit  $N$  Datenpunkten gewählt wird, berechnen sie als

$$p_i = \frac{e^{\frac{-iR}{N}}}{\sum_{n=1}^{N} e^{\frac{-nR}{N}}}, \quad (7.11)$$

wobei  $R$  den *probability decay* bezeichnet. Je größer  $R$ , desto größer ist die Wahrscheinlichkeit, dass schwierige Datenpunkte eher gewählt werden als einfache. Genaue Aussagen werden dazu jedoch nicht getroffen.

Die Kaskade enthält dabei nur ein vollwertiges CNN, das sowohl die erste Ausgabe erzeugt, als auch als Feature Extraktor dient. Die späteren Kaskaden-Teile bestehen



**Abbildung 7.8:** Grafische Darstellung der Kaskadierung, wie sie von Kumar et al. [12] beschrieben wird, mit den CNN-Features und mehreren darauffolgenden FC-Schichten. Quelle: [12].

lediglich aus FC-Schichten, die als Eingabe die CNN-Features sowie die Vorhersagen aller vorherigen Etappen erhalten. Diese Architektur wird schematisch in Abbildung 7.8 dargestellt.

### 7.6.3 Implementierung

Kumar et al. [12] verwendeten ein DenseNet-161 als Backbone und ersetzen die letzte FC-Schicht durch eine, die doppelt so viele Ausgabeneuronen besitzt, wie Label existieren, also 30. Dies begründen die Autoren nicht, wurde aber vermutlich so implementiert, damit für jedes Label sowohl für die positive als auch die negative Klasse ein Ausgabeneuron existiert. Vor der Ausgabeschicht wurde noch eine zweite FC-Schicht eingefügt und zwischen diesen beiden ein Dropout von 0.5 verwendet. Die Autoren berechneten die Softmax-Aktivierung zwischen jeder einzelnen Klasse und dem Rest. Sie verwendeten sechs Modelle für die Kaskade, trainierten mit dem Stochastic Gradient Descent und reduzierten die Lernrate während des Trainings beginnend von 0.1 in zwei Schritten bis zu 0.001.

### 7.6.4 Resultate

Den Autoren gelang es mit ihrer Methode der Boosted Cascade in Kombination mit der Binary Relevance den Stand der Technik für das Label Cardiomegaly zu verbessern, auf den anderen Labeln blieben sie jedoch hinter bisherigen Resultaten zurück.

Sowohl bei der BR als auch bei dem PWE Loss brachte das Boosting Verbesserungen. Nennenswert ist hierbei jedoch, dass die Resultate von Rajpurkar et al. [5] auf allen Labeln außer Cardiomegaly zum Teil erheblich besser ist, obwohl die Methodik dort lediglich in dem Fine-tuning eines auf ImageNet vortrainierten DenseNet-121 besteht. Rajpurkar et al. [5] erzielten beispielsweise für die Label Hernia, Pneumonia und Mass einen AUC der um mindestens 0.05 besser ist, als das beste Ergebnis von Kumar et al. [12]. Für Cardiomegaly ist die AUC von Rajpurkar et al. [5] etwa 0.01 schlechter als das beste Ergebnis von Kumar et al. [12] (Kaskadierung mit BR), aber trotzdem um etwa 0.1 besser als die Kaskadierung mit PWE.

### 7.6.5 Nutzung von Ensembling-Methoden

Durch das Boosting nutzen die Autoren ein Ensembling, das der Mischung der Kombinationen zuzuordnen ist.

## 7.7 Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification

Wie bereits dargelegt wurde, ist die wichtigste Metrik im Bereich der Klassifikation von Thorax-Röntgenbildern die ROC-AUC (vgl. Abschnitt 2.6).

Dies machen sich Yuan et al. [102] zunutze, indem sie eine neue Verlustfunktion vorstellen, die als Stellvertreter-Funktion für den AUC-Wert fungiert. Dies bezeichnen sie als *Deep AUC Maximization* (DAM).

Dafür entwickeln die Autoren den *AUC squared Loss* weiter, den sie als anfällig für Label-Noise und einfache Daten bezeichnen. Die Autoren definieren ihren abstandsbasierte (engl. *margin-based*) AUC Loss über

$$\min_{\substack{w \in \mathbb{R}^d \\ (a,b) \in \mathbb{R}^2}} \max_{\alpha \geq 0} \mathbb{E}_z [F_M(w, a, b, \alpha, z)] \quad (7.12)$$

als Min-Max-Optimierung, wobei

$$\begin{aligned} F_M(w, a, b, \alpha, z) = & (1-p)(h_w(x) - a)^2 \mathbb{I}_{[y=1]} \\ & + p(h_w(x) - b)^2 \mathbb{I}_{[y=0]} - p(1-p)\alpha^2 \\ & + 2\alpha \left( p(1-p)m + ph_w(x)\mathbb{I}_{[y=0]} - (1-p)h_w(x)\mathbb{I}_{[y=1]} \right) \end{aligned}$$

gilt. Dabei bezeichnet  $h_w(x)$  die Modellvorhersage für den Input  $x$  ist und  $y$  das korrekte Label mit der positiven Klasse  $y = 1$  und der negativen Klasse  $y = 0$ . Des Weiteren ist  $p$  die Auftrittswahrscheinlichkeit der positiven Klasse.  $a$  und  $b$  bezeichnen den Erwartungswert der Vorhersage unter der Bedingung, dass die Klasse positiv respektive negativ ist und sind demnach über

$$a = \mathbb{E} [h_w(x)|y = 1], \quad (7.13)$$

---

**Algorithm 1** PESG for solving AUC margin loss

---

**Require:**  $\eta, \gamma, \lambda, T$

- 1: Initialize  $\mathbf{v}_1, \alpha_1 \geq 0$
- 2: **for**  $t = 1, \dots, T$  **do**
- 3:   Compute  $\nabla_{\mathbf{v}} F_M(\mathbf{v}_t, \alpha_t; \mathbf{z}_t)$  and  $\nabla_{\alpha} F_M(\mathbf{v}_t, \alpha_t; \mathbf{z}_t)$ .
- 4:   Update primal variables

$$\mathbf{v}_{t+1} = \mathbf{v}_t - \eta(\nabla_{\mathbf{v}} F_M(\mathbf{v}_t, \alpha_t; \mathbf{z}_t) + \gamma(\mathbf{v}_t - \mathbf{v}_{\text{ref}})) - \lambda \eta \mathbf{v}_t$$

- 5:   Update  $\alpha_{t+1} = [\alpha_t + \eta \nabla_{\alpha} F_M(\mathbf{v}_t, \alpha_t; \mathbf{z}_t)]_+$ .
  - 6:   Decrease  $\eta$  by a factor and update  $\mathbf{v}_{\text{ref}}$  periodically
  - 7: **end for**
- 

Abbildung 7.9: PESG-Algorithmus von Yuan et al. [102] zur Optimierung des DAM.

sowie analog

$$b = \mathbb{E}[h_w(x)|y=0] \quad (7.14)$$

definiert.  $m$  ist ein Hyperparameter für den Abstand zwischen  $a$  und  $b$ .

Für die CheXpert Competition trainierten die Autoren in zwei Stufen. In der pre-training Phase nutzten sie den Cross Entropy Loss, ein Label Smoothing, das an Pham et al. [95] angelehnt ist, den Adam Klassifizierer mit Weight Decay und initialer Lernrate von 1e-5 sowie eine Bildauflösung von  $320 \times 320$ . Als Bildaugmentierung kam zufällige Rotation, Translation und Skalierung zum Einsatz.

Im zweiten Schritt werden die Gewichte der Klassifizierungsschicht zufällig initialisiert bzw. überschrieben. Mit einer initialen Lernrate von 0.1 trainieren sie für 2 Epochen alle Schichten, wobei die Lernrate während des Trainings reduziert wird. In dieser Phase nutzen sie die *proximal epoch stochastic method (PESG)*, den sie von einem Algorithmus zur Optimierung des squared AUC Loss abgewandelt haben, um den DAM Loss zu minimieren. Der PESG-Algorithmus wird in Abbildung 7.9 gezeigt.

Mit einem Ensemble aus DenseNet-121, DenseNet-161, DenseNet-169, DenseNet-201 und einem Inception-resnet--v2 erreichten sie bei der CheXpert Challenge einen mittleren AUC von 0.93 auf dem geheimen Testset und belegen damit zum Zeitpunkt der Entstehung dieser Arbeit den ersten Platz. Auch auf einigen Benchmark-Datensätzen führte der abstandsbasierte AUC Loss in ihren Experimenten zu besseren Ergebnissen als der Cross Entropy Loss oder der squared AUC Loss.

### 7.7.1 Nutzung von Ensembling-Methoden

Die Autoren bilden ein Ensemble mehrerer Modell-Typen, jedoch gehen sie nicht genauer auf die Kombinationsweise ein. Daher scheint es wahrscheinlich, dass ein Soft-Voting zum Einsatz kam.

## 7.8 Modell-Selektion nach AUC

Eine der wichtigsten Metriken bei medizinischen Klassifikationsproblemen, nach der auch die Rangliste der CheXpert Competition sortiert wird, ist die AUC. Daher ist es sinnvoll, jenes Modell zu wählen, das den höchsten AUC-Wert erzeugt. Die Verlustfunktion für das Modell-Training modelliert jedoch für gewöhnlich nicht direkt die Optimierung der AUC. Cortes et al. [103] zeigen, dass die Minimierung der Fehlerrate nicht immer die beste Strategie zur Maximierung der AUC ist. Sie fanden auch heraus, dass Trainingsmethoden, die versuchen, den AUC-Wert direkt zu optimieren, besonders bei unausgeglichenen Distributionen zu besseren Ergebnissen führen können, als die Reduzierung der Fehlerrate.

## 7.9 Intra-Epoch Evaluation

Sowohl aus den Code-Beispielen von [104] als auch von [78], deren Methoden bei der CheXpert Competition die Platzierungen eins und fünf erzielen, lässt sich eine Anpassung des Modell-Trainings finden, die in dieser Arbeit als *Intra-Epoch Evaluation* bezeichnet wird.

Dabei wird das Modell nicht nur am Ende einer Epoche auf den Validierungsdaten evaluiert, sondern alle  $N$  Batches.  $N$  liegt dabei etwa in der Größenordnung 100 – 400. Erzielt das Modell dabei einen neuen Bestwert für die jeweilige Metrik, hier vorzüglich AUC, wird wie beim Epochenende ein Checkpoint der Gewichte erzeugt.

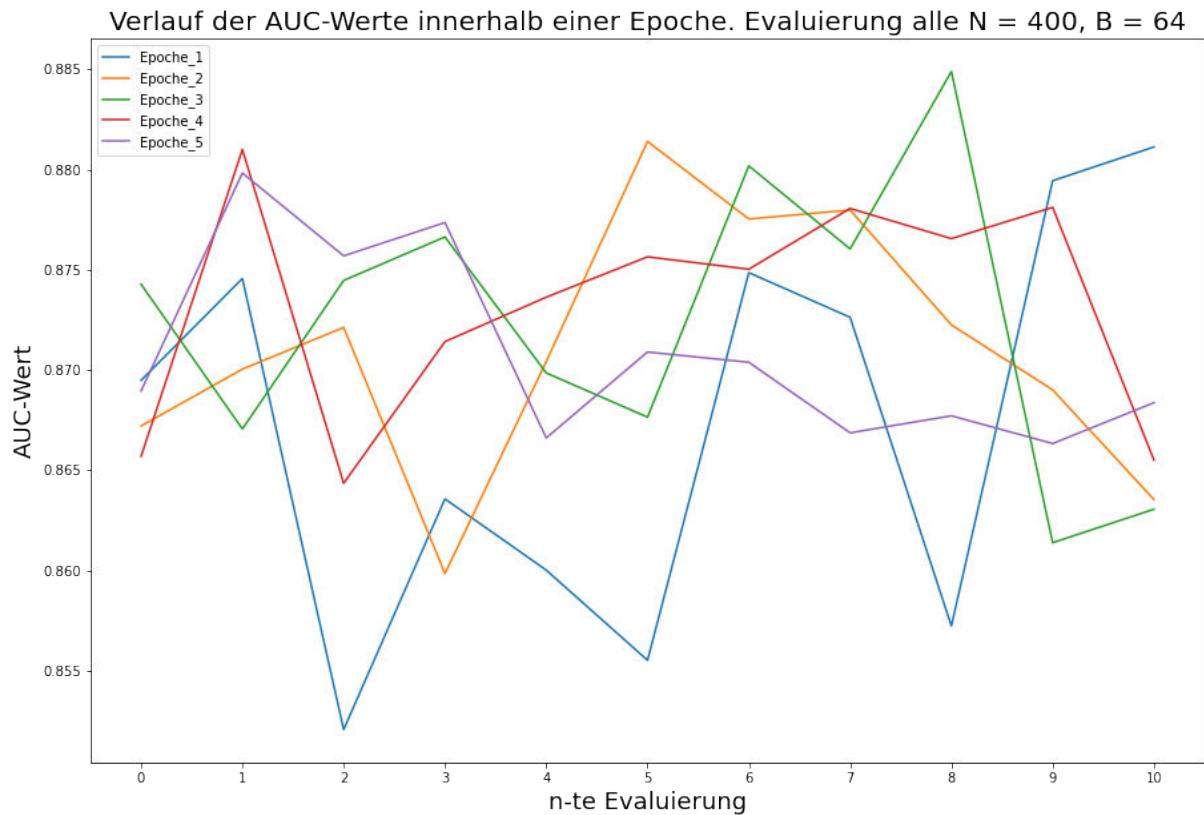
In den Experimenten dieser Arbeit konnte festgestellt werden, dass dieses Vorgehen zu signifikanten Verbesserungen der gewählten Metrik führt. Mehrere Umstände der vorliegenden Problemstellungen begünstigen dies.

Zuerst ist anzumerken, dass der Trainingsdatensatz etwa 1.000 Mal so groß ist wie der Validierungsdatensatz. Zudem sind die Distributionen der Klassen zwischen den Datensätzen unterschiedlich. Diese schwierige Situation lässt sich aufgrund der uncertain Label und dem Anspruch an Goldstandard-Label zur Modellevaluierung nicht auflösen.

Auch die oben beschriebene Modell-Selektion anhand des AUC-Werts sollte in diesem Zusammenhang berücksichtigt werden. Da die Minimierung der Verlustfunktion nicht zwingend zur Maximierung des AUC führen muss [103], kann eine Evaluierung auch bei höheren Loss-Werten sinnvoll sein. Darüber hinaus könnte der negative Einfluss der OoD Bilder, den sie wegen ihrer schlechten Qualität auf das Training haben, durch die Intra-Epoch Evaluation gelindert werden. Denn so werden auch Modell-Zustände evaluiert, die noch nicht mit diesen Bildern trainiert wurden.

Allerdings sollte berücksichtigt werden, dass dies zu einer stärkeren Überanpassung (engl. *Overfitting*) auf dem Validierungsdatensatz führen kann. Außerdem kann die Batchgröße, die meist hardwarebedingt ist, so stärkeren Einfluss auf das Training haben, da sie festlegt, nach wie vielen Trainingsbildern evaluiert wird. Zudem erhöht sich durch die häufigere Evaluation die Trainingsdauer.

Auf den Bildern 7.10 sowie B.14 und B.15 wird der Verlauf der AUC-Werte eines



**Abbildung 7.10:** Beispiel für die Intra-Epoch Evaluation. AUC-Wert, den ein MobileNet V2 auf den CheXpert Validierungsdaten in der jeweiligen Epoche erreicht. Evaluiert wurde alle 400 Batches bei einer Batchsize von 64. Es ist offensichtlich, dass die besten AUC-Werte nicht zwangsläufig am Ende der Epoche erreicht werden und während dieser stark schwanken.

MobileNet V2 auf den Validierungsdaten dargestellt. Dabei wird die Intra-Epoch Evaluation mit  $N \in \{400, 100, 200\}$  verglichen. Diese Beispiele zeigen empirisch auf, dass die Intra-Epoch Evaluation zu relevanten Verbesserungen der AUC führen kann.

# Kapitel 8

## Implementierung

In diesem Kapitel wird die Implementierung der ausgewählten Veröffentlichungen beschrieben. Dabei wird auf Besonderheiten, Schwierigkeiten und Alternativen eingegangen. Dies wird durch ausgewählte Code-Ausschnitte ergänzt, die im Anhang zu finden sind. Zur besseren Lesbarkeit wurden darin typische Import-Befehle wie `import torch` oder `import numpy as np` ausgespart. Die gesamte Code der vorliegenden Arbeit befindet sich auf dem beigefügten Datenträger.

### 8.1 FRODO: Free rejection of out-of-distribution samples: application to chest x-ray analysis

Für die Implementierung von FRODO [91] sind zwei Elemente zentral: Das Extrahieren der Aktivierungen einer bestimmten Schicht sowie die Berechnung der Mahalanobis-Distanz.

Um die Aktivierungen einer Netzsicht zu erhalten gibt es in Pytorch zwei unterschiedliche Ansätze. Der erste besteht darin, die sogenannten Hooks [105] zu verwenden. Diese existieren sowohl für den Forward- als auch den Backward-Pass. Um die Aktivierungen zu erhalten ist nur der Forward-Pass von Interesse, an den ein Hook angebracht werden soll. Die eingehakte Funktion wird am Ende jedes Forward-Passes ausgeführt und sollte die Schicht-Ausgabe in ein Python-Objekt speichern.

Die zweite Möglichkeit besteht in der Modifikation der Modell-Implementierung, sodass diese bei einem Forward-Pass nicht nur den Output des letzten, sondern auch den jener Schicht zurückgibt, deren Aktivierungen betrachtet werden sollen. Diese Implementierung ist demnach etwas weniger flexibel, da der Modell-Code jedes mal angepasst werden muss, wenn die Ausgaben einer anderen Schicht benötigt werden. Dahingegen kann bei dem ersten Ansatz einfach ein neuer Hook an die entsprechende Schicht angefügt werden. Der Vorteil der Code-Modifikation ist, dass sie etwas eleganter ist, da man sich nicht während Training und Inferenz um das Anbringen und Entfernen der Hooks kümmern muss.

Die Modifikation der Modell-Implementierung wird in Abschnitt 8.4 genauer erklärt.

Mit dem Code aus C.1 kann wie von Calli et al. [91] beschrieben das Speichern der Aktivierungen der letzten Convolution in der dritten Schicht (L3) eines ResNet-50 erreicht werden.

Für die Berechnung der Mahalanobis-Distanz können Implementierungen aus `sklearn` [106] genutzt werden. Dort haben beispielsweise die Klassen `ShrunkCovariance` oder `EmpiricalCovariance` Methoden zur Berechnung der quadrierten Mahalanobis-Distanz. Um eine eindimensionale Eingabe zu erhalten wird noch der Mittelwert über die Filter-Dimensionen gebildet. Dies ergibt sich nicht aus [91], wurde aber von den Autoren so durchgeführt, wie sich auf Nachfrage herausgestellt hat.

Beispielsweise hätten die Aktivierungen eines Batches aus der L3-Schicht eines ResNet-50 bei einer Bildauflösung von  $224 \times 224$  die Dimension `batchsize` $\times 512 \times 28 \times 28$ . Der Mittelwert würde in diesem Beispiel über die dritte und vierte Dimension gebildet. Die gemittelten Werte wurden anschließend standardisiert und auf diesen eine Shrunk-Covariance gefittdet.

Bei dem FRODO-Verfahren ist es nötig, einen Schwellwert festzulegen, der über die Zugehörigkeit zur Distribution entscheidet. Um diesen Schwellwert der Mahalanobis-Distanz passend zu definieren, wurden die Lageparameter (arithmetisches Mittel und Standardabweichung) der Distanzen berechnet. Anschließend kann über einen Parameter bestimmt werden, ab wie vielen Standardabweichungen  $s$  Entfernung vom Mittelwert  $\bar{x}$  ein Beispiel nicht mehr zur Distribution gehören soll. Dieses Vorgehen ist inspiriert von der empirischen Regel. Diese besagt dass, wenn eine Normalverteilung  $\mathcal{N}(\mu, \sigma)$  vorliegt, etwa 68% der Daten innerhalb des Bereiches  $\mu \pm \sigma$ , 95% innerhalb von  $\mu \pm 2\sigma$  und 99.7% zwischen  $\mu \pm 3\sigma$  liegen [107]. Daher ist die Regel auch als 68-95-99.7-Regel bekannt und kann zur Anomalieerkennung genutzt werden. Über die Anzahl der Standardabweichungen kann also definiert werden, wie viel Prozent der Daten die korrekte Distribution festlegen sollen.

Nach diesem Vorgehen wurde eine FRODO-Klasse implementiert, mit der OoD-Beispiele detektiert werden können, siehe Listing C.2.

Eine Schwierigkeit bei diesem Ansatz ist, dass zunächst ein möglichst reiner Teildatensatz benötigt wird, um die gewünschte Distribution der Aktivierungen zu definieren. Calli et al. [91] nutzten die Bilder eines bestimmten Labels innerhalb des ChestX-ray14-Datensatzes. Dieses Vorgehen kann jedoch nicht für den vorliegenden Anwendungsfall repliziert werden, da in ChestX-ray14 ausschließlich frontale Bilder vorhanden sind. Eine so erzeugte FRODO würde laterale Bilder als OoD erkennen. Bei Calli et al. [91] war dies von den Autoren explizit erwünscht. Des Weiteren kann die Distribution der ChestX-ray14 Daten selbstverständlich prinzipiell von den CheXpert Aufnahmen abweichen, da diese nicht aus dem gleichen Krankenhaus kommen. Es ist nicht trivial zu sagen, wie resistent die Methodik gegen solche Abweichungen der Domänen beziehungsweise Datenquellen ist.

Daher wurde für den vorliegenden Anwendungsfall die Annahme getroffen, dass die beim CheXpert-Datensatz definierten Validierungsdaten eine hinreichend gute Qualität haben, um die korrekte Distribution vorzugeben, da sie von Radiologen manuell gelabelt wurden.

Das Vorgehen zur OoD-Detektion bestand also darin, zunächst ein ResNet-50 auf

dem CheXpert-Datensatz für 10 Epochen zu trainieren, wobei jenes Modell mit dem niedrigsten Validierungsloss gespeichert wurde. Da gerade die lateralen Bilder häufig gedreht zu sein scheinen, wurde neben dem zufälligen horizontalen Spiegeln auch eine zufällige Rotation mit  $\pm 5$  Grad verwendet.

Im nächsten Schritt wurde ein Forward-Pass mit den designierten Validierungsdaten durchgeführt und dabei die Aktivierungen aufgezeichnet. Diese wurden entlang der Filter-Dimensionen gemittelt und damit ein `FRODO()`-Objekt erzeugt.

Danach wurde erneut ein Forward-Pass über die Trainings-Daten ausgeführt und anhand deren Aktivierungen mit vier Schärfegraden (2, 2.5, 3 und 3.5 Standardabweichungen) über die Zugehörigkeit zur Distribution entschieden.

In Tabelle 8.1 wird dargestellt, wie viele Bilder mit den jeweiligen Standardabweichungen als OoD markiert wurden. Das zeigt, dass eine relativ gesehen geringe Anzahl an Bildern als OoD erkannt wird und keine Normalverteilung vorliegt. Stattdessen liegen die Daten deutlich enger um den Mittelwert, da in  $\bar{x} \pm 2s$  bereits 98% statt 68% der Daten enthält. Damit kann davon ausgegangen werden, dass der Datensatz mit dieser Erkennungsweise weniger OoD-Bilder enthält, als durch eine Normalverteilung erwartet werden würde.

Bei der OoD-Detektion gilt es allerdings auch abzuwegen, wie viele Bilder überhaupt entfernt werden sollen. Schlechte Bilder hindern zwar zu einem gewissen Grad das Training, es sollten aber auch nicht zu viele Bilder entfernt werden um den Datensatz nicht künstlich zu schrumpfen. Schließlich könnten Verzerrungen der Bilder auch als eine Art Augmentierung verstanden werden. Außerdem würde eine große Präsenz solcher Aufnahmen suggerieren, dass deren Qualität durchaus in der Praxis üblich ist. Für etwaige Produktiveinsätze sollten diese deswegen nicht unbedingt abgelehnt werden. Es wäre also medizinisches Fachwissen notwendig, um eine korrekte Distribution festzulegen.

Die Verteilung der Klassen innerhalb der Label der als OoD markierten Bilder ist in Tabelle 8.2 dargestellt. Der Vergleich mit der Labelverteilung des gesamten Datensatzes in Tabelle 5.1 zeigt, dass die Verteilungen sehr ähnlich sind. Dies wäre auch zu erwarten, da man davon ausgehen kann, dass die Bildqualität unabhängig von den Klassen ist. Daher kann dies als Indiz für die Effektivität der FRODO-Methode interpretiert werden.

Die OoD-Information wurde der CSV-Datei des Datensatzes hinzugefügt, sodass die OoD-Bilder vor dem Training herausgefiltert werden können.

Einige Beispiele für Bilder, die mit  $\bar{x} \pm 3.5s$  als OoD markiert wurden, befinden sich in Abbildung 8.1.

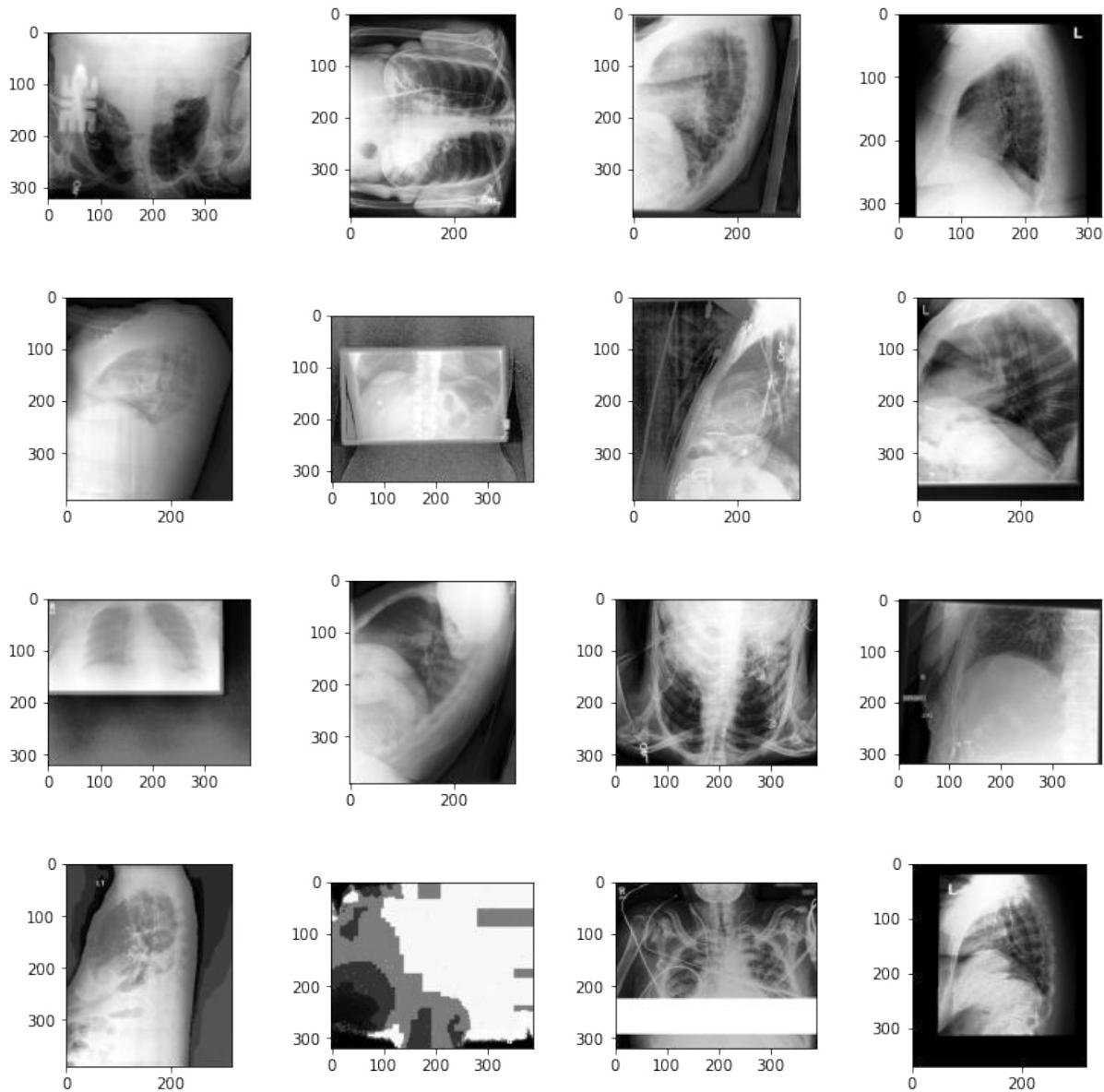
Diese zeigen, dass es durchaus ein gewisses Problem mit der Datenqualität des CheXpert Datensatzes gibt, da sicherlich kein Radiologe mit den zum Teil sehr schlechten Aufnahmen eine Diagnose erstellen würde. Jedoch scheint der Anteil der OoD-Bilder eher gering zu sein und die FRODO-Methode erkannte auch stark rotierte Bilder als OoD, was jedoch diskutabel ist.

Kriterium	In-Distribution (Prozent)	Out-of-Distribution (Prozent)
$\bar{x} \pm 2s$	219060 (98.05%)	4354 (1.95%)
$\bar{x} \pm 2.5s$	221621 (99.20%)	1793 (0.8%)
$\bar{x} \pm 3s$	222562 (99.62%)	852 (0.38%)
$\bar{x} \pm 3.5s$	222936 (99.79%)	478 (0.21%)

**Tabelle 8.1:** Ergebnisse der OoD-Detektion mit FRODO. Es wird die Anzahl und der Anteil der In-Distribution sowie der OoD-Bilder aufgeführt.

Label	Positiv (%)	Negativ (%)	Uncertain (%)
No Finding	448 (10.3 %)	3906 (89.7 %)	0 (0 %)
Enlarged Cardiomediastinum	254 (5.8 %)	3861 (88.7 %)	239 (5.5 %)
Cardiomegaly	594 (13.6 %)	3572 (82.0 %)	188 (4.3 %)
Lung Opacity	1845 (42.4 %)	2395 (55.0 %)	114 (2.6 %)
Lung Lesion	202 (4.6 %)	4108 (94.4 %)	44 (1.0 %)
Edema	789 (18.1 %)	3311 (76.0 %)	254 (5.8 %)
Consolidation	287 (6.5 %)	3553 (81.6 %)	514 (11.8 %)
Pneumonia	161 (3.7 %)	3761 (86.4 %)	432 (9.9 %)
Atelectasis	602 (13.8 %)	3145 (72.2 %)	607 (13.9 %)
Pneumothorax	270 (6.2 %)	4022 (92.4 %)	62 (1.4 %)
Pleural Effusion	1583 (36.4 %)	2523 (57.9 %)	248 (5.7%)
Pleural Other	102 (2.3 %)	4181 (96.0 %)	71 (1.6 %)
Fracture	220 (5.0 %)	4110 (94.4 %)	24 (0.6 %)
Support Devices	1561 (35.9 %)	2760 (63.3 %)	33 (0.8%)

**Tabelle 8.2:** Verteilung der Klassen innerhalb der Label der Bilder, die mit dem Kriterium  $\bar{x} \pm 2s$  als OoD erkannt wurden.



**Abbildung 8.1:** Die ersten 16 als OoD erkannte Bilder des CheXpert-Datensatzes. Einige Bilder weisen eine unbrauchbare Qualität auf (siehe letzte Zeile), es finden sich aber auch Bilder die als False Positive bezeichnet werden können, beispielsweise das dritte Bild in der dritten Zeile, das lediglich vertikal gespiegelt wurde.

## 8.2 Automated Triaging of Adult Chest Radiographs with Deep Artificial Neural Networks

Den Datensatz, den Annarumma et al. [11] für ihr System zur Prioritätsklassifizierung verwenden, ist nicht öffentlich. Zudem soll in dieser Arbeit nur die Krankheitsklassifikation durchgeführt werden und die Einteilung der Label in Dringlichkeitsgruppen ist ohne medizinisches Fachwissen nicht möglich. Basierend auf [11] wurden demnach ein Template Matching (TM) sowie das Ensembling zweier Modelle mit unterschiedlichen

Input-Auflösungen implementiert.

### 8.2.1 Template Matching

Template Matching (TM) wurde nicht nur von Annarumma et al. [11] angewendet, sondern auch von beispielsweise Pham et al. [13] und Baltruschat et al. [16]. Letztere weisen zudem darauf hin, dass das TM in ihren Untersuchungen dazu führte, dass Modelle zu unterschiedlichen Optima konvergieren, was sich positiv auf das Ensemble auswirkt.

Für das TM musste zunächst ein Template festgelegt werden. Dieses wird in Abbildung B.7 dargestellt. Damit wurden verschiedene TM Algorithmen der cv2-Bibliothek mittels eines abgewandelten Codes aus [108] verglichen. Einige Beispiele finden sich in Abbildung 8.2. Die Entscheidung fiel auf den `cv2.TM_SQDIFF_NORMED`-Algorithmus, da dieser am besten zu funktionieren scheint. Da der CheXpert-Datensatz auch laterale Bilder enthält, wurde dafür der Prozess wiederholt. Für beide CheXpert Bildgrößen (Auflösungen  $512 \times 512$  und  $390 \times 320$ , vgl. Abschnitt 5.4) wurde ein eigenes Template aus dem gleichen Bild der jeweiligen Auflösung erzeugt und damit das TM durchgeführt.

Eine Alternative besteht in der Segmentierung der Lunge, wie es beispielsweise Baltruschat et al. [16] mit einem foveal CNN durchgeführt haben, oder unter anderem auch mit der U-Net-Architektur [109] möglich wäre. Diese erfordern allerdings mehr Aufwand als das algorithmische TM.

### 8.2.2 Unterschiedliche Bildauflösungen

Die Implementierung der unterschiedlichen Bildauflösungen besteht lediglich in der entsprechenden Skalierung bei der Datenvorverarbeitung. Die in dieser Arbeit verwendeten Größen sind  $224 \times 224$ ,  $320 \times 320$ ,  $480 \times 480$  und  $512 \times 512$ . Höhere Auflösungen waren aufgrund von Hardwarebeschränkungen nicht möglich.

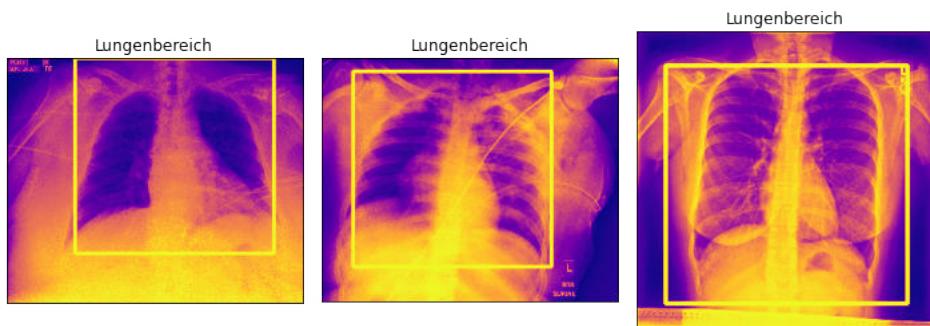
## 8.3 Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels

Die von Pham et al. [95] als am wichtigsten identifizierten Methoden ihrer Arbeit sind die LSR und das Conditional Training. Deren Implementierung wird im Folgenden beschrieben.

### 8.3.1 Label-Smoothing-Regularisierung (LSR)

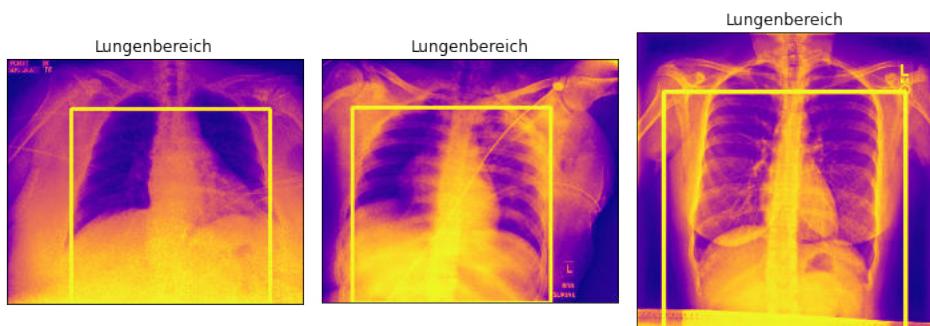
Um die LSR zu realisieren, wurde analog zu den U-Ones und U-Zeros Methoden das uncertain-Label -1 im Datensatz ersetzt. Mittels der Funktion `numpy.random.uniform(low=0.55, high=0.85)` wurden zufällige Werte innerhalb der Grenzen für die U-Ones+LSR Methodik aus [95] erzeugt, da dies bei den Autoren zu den besten Ergebnissen führte.

Methode: cv.TM\_CCOEFF



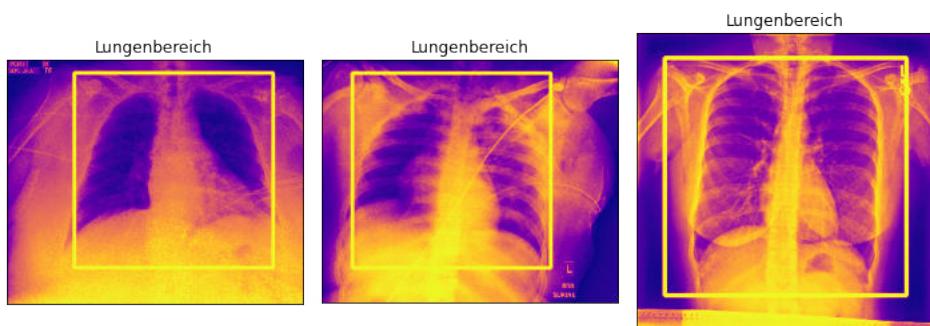
(a) cv2.TM\_CCOEFF-Algorithmus

Methode: cv.TM\_CCORR



(b) cv2.TM\_CCORR-Algorithmus

Methode: cv.TM\_SQDIFF\_NORMED



(c) cv2.TM\_SQDIFF\_NORMED-Algorithmus

**Abbildung 8.2:** Vergleich dreier Template Matching Algorithmen. Zur besseren Erkennbarkeit der Lunge wurde die Plasma-Colormap verwendet. Die Bilder stammen aus dem CheXpert-Datensatz von den Patienten 1, 11 und 21, jeweils Studie 1 und zeigen view1\_frontal

### 8.3.2 Conditional Training

Für das Conditional Training wurde der Datensatz nach Bildern gefiltert, bei denen die beiden Eltern-Pathologien Lung Opacity und Consolidation positiv sind. Darauf wurden die verschiedenen Modellarchitekturen vortrainiert und anschließend alle Schichten bis auf die letzte eingefroren, sodass nur die Klassifizierungsschicht trainiert wird. Im nächsten Schritt wurde das Fine-tuning auf dem gesamten Trainingsdatensatz durchgeführt. Neben dieser genauen Umsetzung der Methodik von Pham et al. [95] ist es auch durchaus denkbar bei dem Fine-tuning abzuweichen und mehrere oder gar keine Schichten einzufrieren.

## 8.4 Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification

Das AG-CNN besteht aus einem Backbone-Modell (DenseNet oder ResNet), der sowohl für den Global Branch und den Local Branch verwendet wird, sowie aus dem Fusion Branch, der aus einer einzelnen FC-Schicht besteht. Die Backbone-Modelle sind bereits in der `torchvision`-Bibliothek (vgl. Kapitel A.1) implementiert.

Die Besonderheiten der Implementierung liegen hauptsächlich in der Erzeugung der Masken aus den Heatmaps. Sowohl hier, als auch für den Input des Fusion Branches, ist es nötig, die Ausgaben gewisser Schichten aus dem Backbone-Modell zu extrahieren. Dies könnte zunächst wie in Abschnitt 8.1 beschrieben über Hooks umgesetzt werden, was für den ResNet-Backbone auch so durchgeführt wurde. Für den DenseNet-Backbone wurde die Modell-Implementierung modifiziert, sodass sowohl der Output der letzten Conv-Schicht für die Heatmaps als auch die Ausgabe der letzten Pooling-Schicht für den Fusion Branch bei einem Forward-Pass zurückgegeben werden können. Die Unterscheidung in den Herangehensweisen nach Backbone ist dabei willkürlich gewählt und könnte auch beliebig anders ausfallen.

Die DenseNet-Architektur ist im Bibliotheksmodul `torchvision.models.densenet.py` implementiert. Darin gibt es die Klasse `DenseNet(nn.Module)`, deren `forward()`-Methode modifiziert wurde. Es wurden Parameter hinzugefügt, über die die Rückgabe von Convolution- und Pooling-Output gesteuert werden kann. Ferner wurden diese Ausgaben in eigene Variablen gespeichert, um sie nicht zu überschreiben und die Rückgabe zu ermöglichen. Die angepasste `forward()`-Methode befindet sich in Listing C.3.

Bei der Trainingsreihenfolge wurde der sequenziellen G\_L\_F-Strategie gefolgt, die bei Guan et al. [100] am erfolgreichsten war. Für das Erzeugen der Heatmaps wird zunächst das Maximum über den Absolutbetrag der Aktivierungen der letzten Convolution-Schicht des Global Branches ermittelt und dies dann auf die Bildauflösung skaliert. Um die Skalierung ohne `torchvision` oder beispielsweise das Paket `PIL` und dadurch notwendiges Casting durchzuführen, wurde diese mittels der `torch.nn.functional.interpolate()`-Funktion implementiert. Die umgesetzten Funktionen sind in Listing C.4 aufgeführt.

Anschließend müssen aus den Heatmaps die Masken über Thresholding und der

Identifizierung der größten verbundenen Region generiert werden. Dafür wurde die `skimage`-Bibliothek genutzt und das Beispiel [110] als Grundlage verwendet. Die Funktion `getLargestConnectedAreaMask()` nimmt eine einzelne Heatmap entgegen und nutzt entweder einen als Parameter übergebenen Threshold, den Otsu-Threshold oder  $\bar{x} + s$  als Schwellwert. In einem Vorverarbeitungsschritt wird noch auf den Wertebereich  $[0, 1]$  normalisiert, was die Übergabe eines Threshold-Wertes vereinfacht. Dieser Schwellwert wird angewendet, dann durch die `skimage.morphology.closing()`-Funktion kleine Löcher gefüllt, Artefakte am Rand entfernt und dann werden die vorhandenen Bildregionen über die Funktion `skimage.measure.label(cleared, return_num=True)` identifiziert. Durch den Parameter `return_num=True` wird die Anzahl der Regionen zurückgegeben. Dies hat den Vorteil, dass die größte verbundene Region gleichzeitig den höchsten Index, also die Anzahl der Regionen, besitzt. Somit lässt sich die größte Region einfach filtern. Diese stellt den gesuchten, vom Global Branch fokussierten, Bereich dar. Der Code befindet sich in Listing C.5.

Daraufhin werden diese Masken verwendet um aus dem globalen Bild die lokale ROI auszuschneiden. Für die Skalierung kann erneut die Funktion `resizeHeatmapsBatch()` genutzt werden. Falls die Maske leer ist, weil beispielsweise kein Bereich über dem Schwellwert lag oder ein anderes Problem aufgetreten ist, wird das Originalbild verwendet. Dies wird in Listing C.6 gezeigt.

## 8.5 A Novel Approach for Multi-Label Chest X-Ray Classification of Common Thorax Diseases

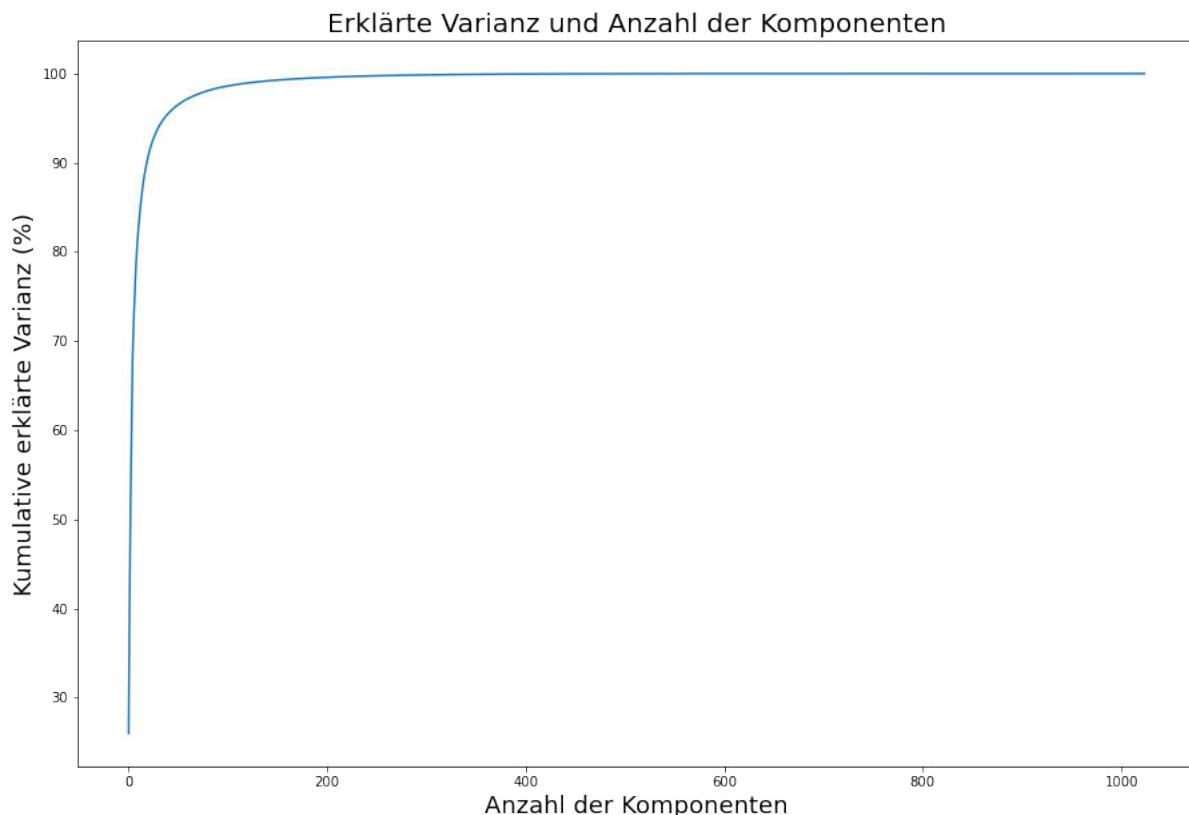
Der erste Schritt dieser Methode ist das Training eines CNNs wofür zunächst ein DenseNet-121 genutzt wurde. Anschließend beschreiben Allaouzi et al. [52], dass sie die Klassifizierungsschicht entfernen. Dies wurde so umgesetzt, dass diese nicht entfernt wird, sondern durch eine Identitätsschicht (vgl. Listing C.7) ersetzt wurde, die den Input unverändert zurückgibt. Alternativ wäre es auch möglich gewesen Hooks zu verwenden oder die Modell-Implementierung anzupassen (vgl. Abschnitte 8.1 und 8.4). Für die Problemtransformationen wurde die Bibliothek `scikit-multilearn` [111] verwendet. Diese ist mit `scikit-learn`-Klassifizierern kompatibel. Beispielsweise kann eine BR-Transformation mit einer Logistischen Regression einfach als `skmultilearn.problem_transform.BinaryRelevance(sklearn.linear_model.LogisticRegression())` definiert werden. Dieses Objekt bietet dann die gewohnten `fit()` und `predict_proba()` Methoden an.

Derartige Objekte wurden für die verschiedenen Problemtransformationsmethoden erzeugt und Trainingsdaten generiert, indem mit dem CNN mit Identitätsschicht ein Forward-Pass für die Trainingsbilder durchgeführt wurde.

Anders als bei Allaouzi et al. [52] wurde zudem auf den Eingaben für die Klassifizierer eine Dimensionsreduktion durch eine Principle Component Analysis (PCA) erprobt. Dafür erfolgte zunächst mittels der Klasse `sklearn.preprocessing.StandardScaler` eine Standardisierung der Feature. Die PCA-Implementierung aus `sklearn.decomposition.PCA` führte aufgrund der sehr großen Dimension der Daten allerdings zu einem Absturz des

Python-Interpreters. Daher wurde die PCA wie in [112] beschrieben über die Funktion `numpy.linalg.eig()` implementiert. Diese Realisierung basiert auf der Eigenzerlegung der Kovarianzmatrix  $C_x^{-1}$  und ist dadurch deutlich ressourceneffizienter. Aufgrund numerischer Instabilität bei den verwendeten Gleitkommazahlen, entstanden dabei komplexe Zahlenteile, die allerdings mittels `numpy.real()` verworfen werden können, da sie ohnehin fast vollständig 0 entsprechen. Über die so erhaltenen Eigenwerte kann untersucht werden, wie stark die Dimensionalität reduziert werden kann, ohne zu viele Informationen zu verlieren. Die Informationen entsprechen in diesem Fall dem Anteil der erklärten Varianz. In Abbildung 8.3 wird die erklärte Varianz kumulativ visualisiert. Dieser Plot zeigt, dass bereits mit einer geringen Hauptkomponentenanzahl ein großer Anteil der Information abgebildet werden kann. Der Vergleich in Tabelle 8.3 offenbart, dass die ersten 512 Hauptkomponenten nahezu die gesamte Information enthalten. Für die weiteren Experimente fiel die Entscheidung, die ersten 36 bzw. 50 Hauptkomponenten und somit 95% bzw. 96.4% der Information zu behalten. Die Dimension der  $N$  Trainingsdaten für die Klassifizierer reduziert sich somit von  $N \times 1024$  auf  $N \times 36$  respektive  $N \times 50$ .

Diese Untersuchung deutet an, dass die Repräsentation der Feature durch das CNN nicht sehr effizient ist und für den vorliegenden Anwendungsfall mit deutlich weniger als den von der Architektur verwendeten 1024 Neuronen auskommen könnte.



**Abbildung 8.3:** Kumulative Darstellung der erklärten Varianz der  $n$  Hauptkomponenten der Eigenzerlegung. Als Feature Extractor wurde ein DenseNet-121 genutzt.

Anzahl Hauptkomponenten	Erklärte Varianz (kumulativ)
1	26%
2	41%
3	52%
36	95%
50	96%
124	99%
512	99,9%

**Tabelle 8.3:** Vergleich der erklärten Varianz je nach Anzahl der gewählten Hauptkomponenten.

## 8.6 Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs

Für die Kaskadierung wurde eine Klasse implementiert, die als Wrapper-Klasse für das Backbone-CNN dient und neben der Vorhersage auch die Ausgabe der letzten Schicht vor der Klassifizierung zurückliefert. Für die darauffolgenden Kaskaden-Elemente wurde ebenfalls eine Klasse entwickelt, die die FC-Schichten in Abhängigkeit ihrer Position innerhalb der Kaskade korrekt initialisiert.

Der PWE-Loss kann in einer „naiven“ Implementierung realisiert werden, indem Gleichung (7.9) iterativ umgesetzt wird. Dabei wird über jedes Batch-Element iteriert, Formel (7.9) angewendet und zum Loss addiert. Schließlich kann analog zu den in Pytorch implementierten Verlustfunktionen der Mittelwert gebildet werden. Diese Implementierung befindet sich in Listing C.9.

Eine geschicktere Umsetzung findet sich in dem Code von [113]. Der Autor verwendet dabei den Broadcasting-Mechanismus, um die paarweisen Differenzen des Inputs durch die `unsqueeze()`-Funktion als Matrix zu berechnen. Das gleiche Prinzip wird erneut verwendet, um die Indizes zu finden, an denen die Label paarweise unterschiedlich sind. Dies ist gleichbedeutend mit dem iterieren über  $u$  und  $v$ , da  $u \in \mathcal{Y}_i$  und  $v \notin \mathcal{Y}_i$  ist. Eine kommentierte Version dieser Implementierung befindet sich in Listing C.10. Der Geschwindigkeitsvorteil liegt dabei bei einem Faktor von über 100 (vgl. Abbildung B.8).

Von der Methode von Kumar et al. [12] wurde dahingehend abgewichen, dass lediglich die Kaskadierung und der PWE Loss übernommen wurden. Die BR und Einsatz von Softmax- statt Sigmoid-Aktivierungsfunktion wurde nicht realisiert. Außerdem wurde mit dem Adam-Optimisierer mit geringerer Lernrate trainiert, was aufgrund der übrigen Erfahrungen sinnvoll schien. Statt schwierigere Bilder eher zu samplen als einfache, wurde deren Gewichtung bei der Berechnung der Verlustfunktion angepasst.

## 8.7 Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification

Die beiden zentralen Elemente der Methode von Yuan et al. [102] sind der abstands-basierte AUC-Loss (DAM) sowie der zugehörige Optimisierer PESG. Die Autoren veröffentlichten die Bibliothek LibAUC [104], in der diese implementiert sind.

Das Training wird in zwei Schritten durchgeführt. Im ersten wird das Modell mit dem Cross-Entropy-Loss auf den fünf Vergleichslabellen trainiert. Anschließend wird die letzte Schicht so angepasst, dass sie nur noch ein Ausgabeneuron besitzt, auf das eine Sigmoid-Aktivierung folgt. Von dieser Modell-Vorlage werden fünf Kopien erzeugt, die dann auf je eines der Vergleichslabel mittels LibAUC trainiert werden. Dieser Schritt muss vorgenommen werden, da der DAM-Loss zunächst nur für single-label Klassifikationen implementiert war. Die Möglichkeit zur Multi-Label Klassifikation wurde erst später hinzugefügt. In beiden Trainingsetappen wird dabei die Intra-Epoch Evaluation eingesetzt. Die Dataset-Klasse aus der LibAUC-Bibliothek wurde für das Training angepasst. Das Laden der Bilder und deren Vorverarbeitung wurde an die bisherige Implementierung angeglichen sowie die Unterstützung mehrerer Behandlungen der uncertain Label hinzugefügt. Um die Anwendung handlicher zu gestalten wurde die Hilfsklasse DAUCHelper implementiert.

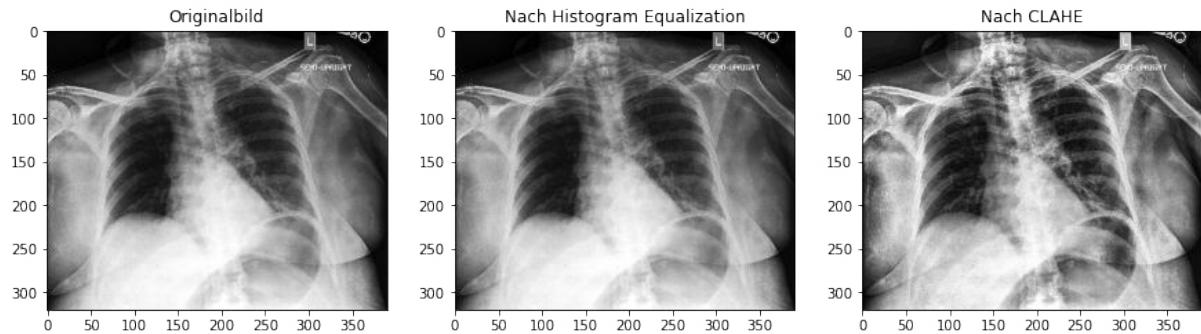
## 8.8 Weitere Implementierungen

Neben den Implementierungen zu den ausgewählten Methoden wurde auch die Contrast Limitation Adaptive Histogram Equalization (CLAHE) als Erweiterung der HE umgesetzt, sowie verschiedene Ansätze zum Ausgleichen der Imbalance des Datensatzes analysiert. Dies wird im Folgenden beschrieben.

### 8.8.1 CLAHE

Dunnmon et al. [74] verwenden eine Histogram Equalization (HE). Eine Erweiterung dazu ist die Adaptive HE, bei der mehrere Bildausschnitte einzeln ausgeglichen werden [114]. Die sogenannte CLAHE führt zusätzlich ein Clipping ein, wobei hohe Ausschläge im Histogramm abgeschnitten und auf den Rest des Histogramms verteilt werden [114]. Solche Peaks stammen aus Bildbereichen mit nahezu gleichmäßiger Verteilung. Ohne das Clipping kann dies dazu führen, dass Noise im Bild übermäßig betont wird, da der schmale Bereich von Eingabe-Graustufen der hohen Ausschläge auf einen großen Ausgabe-Graustufen-Bereich abgebildet würde [114]. Wichtige Parameter der CLAHE sind daher das Clipping-Limit sowie die Größe der Bildausschnitte.

Die CLAHE ist kurzum eine verbesserte Methode zur HE, die daher für diese Arbeit genutzt wurde. Ein Beispiel für den Effekt findet sich in Abbildung 8.4. Dieses zeigt, dass die Kontraste deutlich stärker sind und die Lunge sich leichter von den Knochen unterscheiden lässt.



**Abbildung 8.4:** Beispiel für den Effekt der Globalen HE sowie der CLAHE mit Clipping-Limit 2.0 und Bildausschnittsgröße  $8 \times 8$ . Das Originalbild stammt aus dem CheXpert-Datensatz, Patient 3, Studie 1, Bild 1. Dieser Vergleich verdeutlicht, dass die normale HE kaum Auswirkungen zeigt, während durch die CLAHE beispielsweise das Skelett deutlicher erkennbar ist.

### 8.8.2 Behandlung der Daten-Imbalance

Wie in Tabelle 5.1 und in den Abbildungen 5.3 bzw. B.6 auf den ersten Blick zu sehen ist, sind die Label des CheXpert-Datensatzes stark unausgeglichen. Auch innerhalb der Label sind die Klassen unausgeglichen, wie aus Abbildung B.9 ersichtlich wird.

Eine solche Imbalance kann die Generalisierungsfähigkeit eines Klassifizierers behindern, da die Merkmale der seltenen Klassen schlechter gelernt werden können [115]. Auf der Datenebene lässt sich ein Resampling durchführen, um den Datensatz auszugleichen. Das Resampling lässt sich in hauptsächlich zwei Arten unterteilen: beim Oversampling werden die seltenen Klassen häufiger gezogen (engl. *oversampled*) und analog werden beim Undersampling die häufigen Klassen seltener gezogen [115].

Dabei ist es nicht zwingend so, dass ein derartiges Ausgleichen der Daten, sodass die Klassen in der gleichen Häufigkeit vorkommen, zu den besten AUC-Werten führen muss, sondern die beste Verteilung ist datensatzspezifisch [115]. Beim Oversampling von Bilddaten bietet es sich an, die Datenpunkte nicht nur zu ziehen, sondern auch Bildaugmentierung anzuwenden (vgl. Abschnitt 2.3), wobei diese selbstverständlich die Labelkorrektheit beibehalten sollte.

Eine Alternative zum Resampling kann ein kosten-sensitives Lernen sein [115]. Die Fehlklassifikation von seltenen Klassen sollte härter bestraft werden, um ihre geringere Auftrittshäufigkeit auszugleichen. Dies lässt sich über eine Gewichtung realisieren. Die in Pytorch implementierten Verlustfunktionen bieten dafür den Parameter `weight` an, bei dem für jedes Batch-Element eine Gewichtung angegeben werden kann. Eine weitere Möglichkeit besteht in der Nutzung des `pos_weight`-Parameters. Mit diesem kann die Gewichtung der positiven Klasse gegenüber der negativen Klasse angepasst werden.

Außerdem kann mit dem Boosting eine Ensembling-Technik angewendet werden (vgl. AdaBoost in Kapitel 2.5.3) oder der Klassifizierungsalgorithmus in anderer Weise angepasst werden um eine Klassenimbalance zu entschärfen [115].

Die Synthetic Minority Over-sampling Technique (SMOTE) von Chawla et al. [116]

ist eine spezielle Oversampling-Methode, bei der Datenpunkte zum Oversampling nicht einfach dupliziert werden, sondern neue, synthetische Datenpunkte der Minderheitsklasse erzeugt werden. Dafür wird ein k-NearestNeighbors Klassifizierer genutzt, der einige nahe gelegene Datenpunkte identifiziert. Der Feature-Vektor eines davon zufällig gewählten Nachbarn kann dann mit einer zufälligen Zahl zwischen 0 und 1 multipliziert werden und das Ergebnis auf den ursprünglichen Feature-Vektor addiert werden, um einen synthetischen Datenpunkt zu erzeugen. Dieser liegt zwischen den beiden Nachbarn gleicher Klasse und die Decision Boundary sollte ihn somit einschließen, wodurch sichergestellt ist, dass dem synthetischen Datenpunkt bedenkenlos die Minderheitsklasse zugeordnet werden kann.

Die Multilabel Synthetic Minority Over-sampling Technique (MLSMOTE) von Charte et al. [47] ist eine Erweiterung der SMOTE-Methode auf Multilabel-Probleme. Die Multi-Label Klassifikation bringt dabei das Problem mit sich, dass ein Datenpunkt gleichzeitig einer Mehrheits- und einer Minderheitsklasse zugeordnet sein kann. Anhand von Abbildung 5.3 lässt sich feststellen, dass dies unter anderem zwischen den Labeln Support Devices und Lung Lesion sowie Pneumothorax auftritt. Dies lässt sich dadurch erklären, dass Support Devices unabhängig von der vorhandenen Pathologie häufig zutrifft, wodurch sich für dieses Label eine große Datenmenge ergibt, während die einzelnen Pathologien deutlich seltener sind.

Bei der MLSMOTE-Methode werden daher zwei Metriken berechnet, um Minderheitslabel zu identifizieren: die  $IRLbl$ , die für jedes Label die Imbalance misst und  $MeanIR$ , die der Mittelwert der einzelnen  $IRLbl$  ist [47]. Berechnet wird

$$IRLbl(l) = \frac{\operatorname{argmax}_{l'=\mathcal{L}_1}^{\mathcal{L}_{|\mathcal{L}|}} \left( \sum_{i=1}^{|D|} h(l', Y_i) \right)}{\sum_{i=1}^{|D|} h(l, Y_i)}, \quad h(l, Y_i) = \begin{cases} 1, & \text{wenn } l \in Y_i, \\ 0, & \text{wenn } l \notin Y_i, \end{cases} \quad (8.1)$$

wobei  $D$  der Multilabel-Datensatz ist,  $\mathcal{L}$  alle Label darstellt,  $\mathcal{L}_l$  das  $l$ -te Label ist und  $Y_i$  das korrekte Labelset, das dem  $i$ -ten Datenpunkt zugeordnet ist [47].

Die  $MeanIR$  berechnet sich als arithmetischer Mittelwert über die Formel  $MeanIR = \frac{1}{|\mathcal{L}|} \sum_{l=\mathcal{L}_1}^{\mathcal{L}_{|\mathcal{L}|}} IRLbl(l)$  [47].

Die Minderheitslabel werden daran erkannt, dass für sie gilt  $IRLbl(l) > MeanIR$  [47]. Anschließend können ähnlich zu dem SMOTE Vorgehen über Distanzberechnung Nachbarn gewählt und synthetische Datenpunkte erzeugt werden.

Da in der vorliegenden Arbeit allerdings Bilddaten verwendet wurden, liegen kein direkten Feature-Vektoren vor. Diese werden erst von den CNNs erzeugt. Daher wäre sowohl die Anwendung eines kNN-Algorithmus zur Identifizierung von Nachbarn also auch die anschließende Generierung von synthetischen Feature Vektoren fragwürdig. Es könnte versucht werden, ein CNN als Feature Extractor zu verwenden (vgl. Abschnitt 7.5) und anschließend den MLSMOTE-Algorithmus anzuwenden, dieses Verfahren birgt aber viele Unsicherheiten.

In dieser Arbeit wurde jedoch ein der MLSMOTE-Algorithmus nur verwendet, um die Minderheitslabel zu erkennen. Anschließend wurden die so identifizierten Bilder dem

Datensatz erneut hinzugefügt. Die während dem Training verwendete Bildaugmentierung ersetzt dann das synthetische Erzeugen von Datenpunkten.

Beim MLSMOTE-Algorithmus ist es auch wichtig zu berücksichtigen, dass die  $IRLbl$  nach jeder Iteration, in der Datenpunkte erzeugt wurden, neu berechnet wird. Um dies Nachzubilden muss das Vorgehen dieser Arbeit mehrmals wiederholt werden.

Der Trainingsdatensatz wurde in der ersten Iteration um etwa 85.000 Datenpunkte auf insgesamt rund 309.000 Bilder erweitern. Das Chord-Diagramm dazu befindet sich in Abbildung B.10. Allerdings konvergiert dieses Vorgehen nicht und der Datensatz wird mit den Iterationen immer größer, während sich die Ausgeglichenheit der Klassen nur leicht verbessert. Nach vier Iterationen hätte der Datensatz schon über eine Millionen Datenpunkte.

Da dieses Vorgehen sich also als nicht hilfreich erwies, wurde der Datensatz weiter analysiert und ein Oversampling von bestimmten Labelsets durchgeführt. Da normalerweise je Krankheit die positive Klasse unterrepräsentiert ist und die uncertain Label meist nur einen geringeren Teil ausmachen, wurde als Basis das U-Ones Vorgehen gewählt. Die Verteilung der Klassen je Label, die sich daraus ergibt, befindet sich in Abbildung B.11. Aus der Analyse der dann vorhandenen Labelsets ergaben sich die folgenden, zentralen Erkenntnisse.

Es existieren 2434 einzigartige Labelsets, was etwa 10% der Daten entspricht. 2159 davon besitzen maximal 100 Beispiele. Für den größten Teil der Labelkombinationen gibt es also nur sehr wenige Aufnahmen, denen diese zugeordnet sind. Das häufigste Labelset ist {No Finding}, das ungefähr 13.500 Mal vorkommt. Dies wird dadurch begünstigt, dass No Finding der Logik nach keine Kookkurrenz mit Pathologien, sondern höchstens mit dem Label Support Devices, haben sollte. Im anderen Extrem treten 467 Labelsets nur ein einziges Mal auf.

Hervorzuheben ist, dass bei 4256 Bildern alle Label (trotz U-Ones) negativ sind. Hier liegt zwingend ein Fehler des Labeling-Tools vor, da bei Abwesenheit aller Pathologien zumindest No Finding positiv sein müsste. Der Fehler kann jedoch nicht korrigiert werden, da die zugehörigen Radiologieberichte nicht öffentlich sind. Dies entspricht etwa 20 mal der Bildanzahl der Validierungsdaten. Darauf basierend wurden die knapp 4.300 fehlerhaften Bilder markiert, um sie entfernen zu können. Anschließend wurden die Bilder für spezifische Labelsets herausgefiltert und mehrfach dem Datensatz hinzugefügt, um die positiven Klassen ausgeglichener zu gestalten. Insgesamt wurden so rund 150.000 duplizierte Bilder hinzugefügt, sodass der erweiterte Datensatz 371.945 Zeilen enthält. Die daraus resultierende Verteilung ist in Abbildung B.12 dargestellt. Zum Teil gab es aber sehr wenig Beispiele je Labelset, sodass hohe Multiplikatoren angewendet werden mussten. Dies kann durchaus als Problem dieses Vorgehens identifiziert werden, auch wenn es durch stärkere Bildaugmentierung etwas gelindert werden kann. Eine Möglichkeit zur Erweiterung bestünde darin, die Bilder bereits vor dem Hinzufügen zum Datensatz zu augmentieren. In dieser Arbeit wurde aber nur auf die Augmentierungen während des Trainings gesetzt.

Neben dem Oversampling basierend auf den Labelsets ist es auch möglich, die Gewichtung eines Datenpunktes für den Loss anhand des Labelsets zu berechnen. Die Gewichtung  $w_x$  eines Datenpunktes  $x$  mit dem Labelset  $l_x$  aus dem Datensatz  $D$  mit

Länge  $|D|$  wurde aus der Auftrittshäufigkeit  $|l_x|$  als

$$w_x = 1 - \frac{|l_x|}{|D|} \quad (8.2)$$

berechnet. Somit werden Bilder, deren Labelset seltener auftritt, höher gewichtet. Eine Fehlklassifikation würde dann stärker durch den Loss bestraft werden.

# Kapitel 9

## Durchgeführte Experimente

Das folgende Kapitel zeigt die Vorbereitungen für die Experimente sowie den Aufbau dieser auf. Alle Versuche wurden auf einem Linux-System mit einem AMD Ryzen 9 3900X Prozessor, 64 Gigabyte Arbeitsspeicher und einer GeForce RTX 2080 SUPER mit 8 Gigabyte Speicher durchgeführt.

### 9.1 Vorbereitungen

Bevor die Experimente durchgeführt werden konnten, wurden einige Vorbereitungen getroffen, die im Folgenden kurz dargestellt werden sollen.

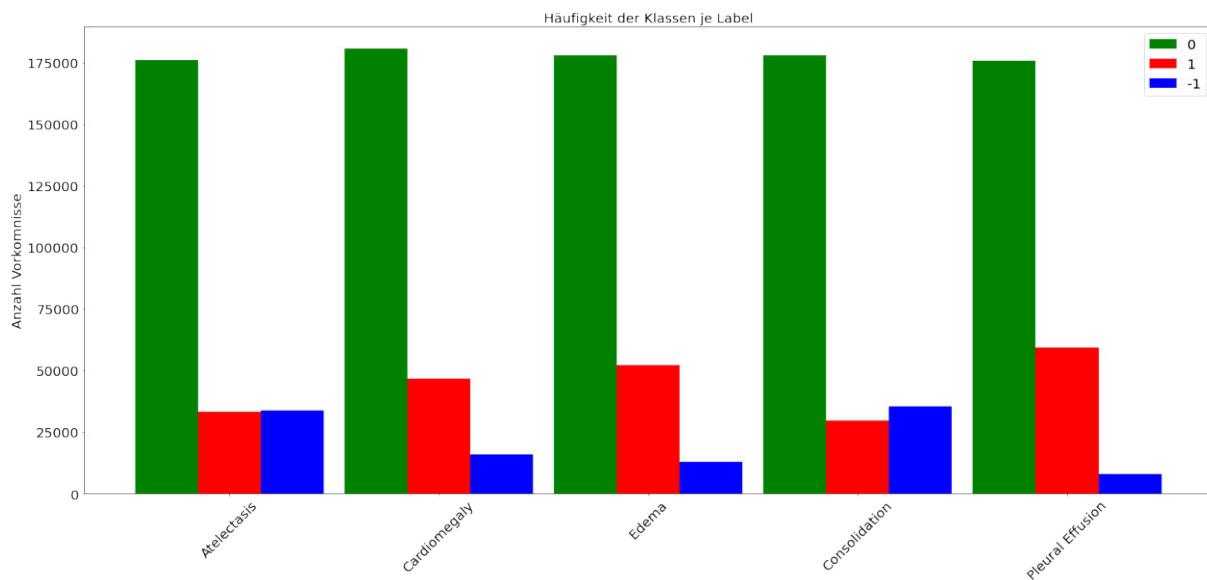
#### 9.1.1 Datensatz ausbalancieren

Wie in Abschnitt 8.8.2 beschrieben, wurde zunächst versucht, den Datensatz durch Oversampling von Bildern, denen seltene Labelsets zugeordnet sind, auszubalancieren. Darüber hinaus wurde ein ähnliches Vorgehen auch für ausschließlich die fünf Vergleichslabel durchgeführt. Dabei wurden die uncertain Werte beibehalten. Die Verteilung der Klassen für die Vergleichslabel wird in Abbildung B.13 und Tabelle 9.1 visualisiert. Betrachtet man für diese die Summe aus der positiven und der uncertain Klasse, scheint es sinnvoll, ein Ziel von in Summe 65.000 Bildern festzulegen. Besonders da es rund 30.000 Bilder gibt, bei denen nur Pleural Effusion nicht-negativ ist, und alle anderen Vergleichslabel negativ sind, ist diese Zahl naheliegend. Denn diese rund 30.000 Bilder können entfernt werden, ohne die Verteilung der anderen Label zu beeinflussen.

Zunächst wurde daher das Label Pleural Effusion in dieser Form undersampled. Anschließend wurden für die Label Cardiomegaly und Consolidation die Bilder gefiltert, die ausschließlich dort positiv sind, und diese oversampled. Auf diese Weise konnte der Datensatz so ausgeglichen werden, dass die fünf Vergleichslabel (nahezu) gleich häufig nicht-negativ vorkommen (vgl. Abbildung 9.1). Die Gesamtlänge des Datensatzes wird bei diesem Vorgehen nur gering auf 243.091 Bilder erhöht. Jedoch führt dies

Label	Positiv	Uncertain	Summe	Maßnahme
Atelectasis	33376	33739	67115	Keine
Cardiomegaly	27000	8087	35087	Oversampling
Edema	52246	12984	65230	Keine
Consolidation	14783	27742	42525	Oversampling
Pleural Effusion	86187	11628	97815	Undersampling

**Tabelle 9.1:** Ausgangssituation und daraus resultierende Maßnahme für das Ausbalancieren der Vergleichslabel.



**Abbildung 9.1:** Verteilung der Klassen je Label für die fünf Vergleichslabel nach ihrem Ausbalancieren. Die nicht-negativen Klassen sind zwischen den Labels ausgeglichen.

selbstverständlich dazu, dass der gesamte Datensatz weiterhin nicht balanciert ist, wie in Abbildung 9.2 ersichtlich wird.

### 9.1.2 Datensatz an Validierungsdaten angleichen

Neben den zwei Ansätzen zum Ausbalancieren des Datensatzes wurde noch eine dritte Modifikation durchgeführt: Statt die Klassen bzw. Label auszugleichen, wurde deren Verteilung an die des designierten Validierungsdatensatzes angeglichen. Da dieser keine uncertain Label enthält, musste für die uncertain Label des Datensatzes erneut entschieden werden, wie diese behandelt werden sollen. Wie bei den vorherigen Anpassungen wurde der U-Ones Ansatz gewählt, bei dem uncertain als positiv aufgefasst wird. Der Vergleich der Verteilungen in Tabelle 9.2 zeigt, dass es zwischen den Trainings- und Validierungsdaten durchaus Unterschiede gibt. Zunächst wurden die Bilder, die sowohl bei Pleural Effusion als auch Edema nicht-negativ sind, aus den Trainingsdaten entfernt, da diese Kombination in den Validierungsdaten nicht vorkommt. Anschließend wurden für diese beiden Label noch weitere Bilder zufällig

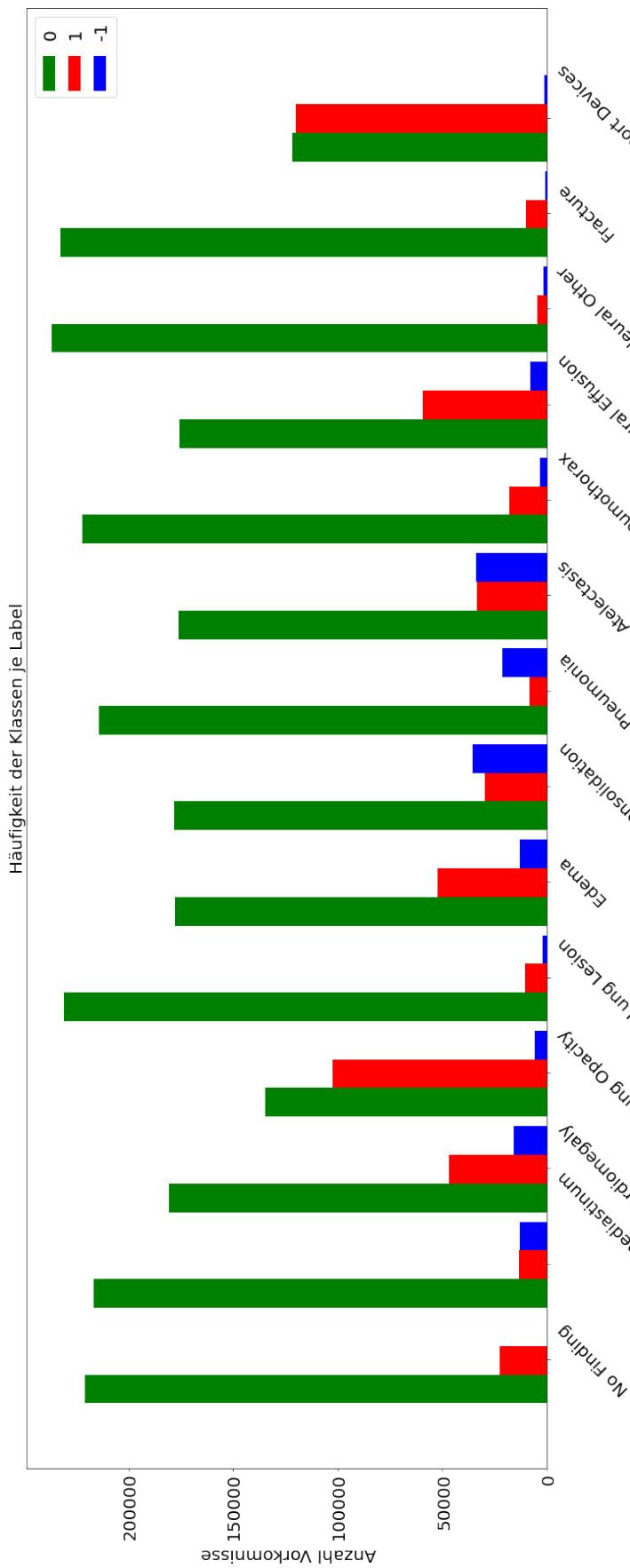
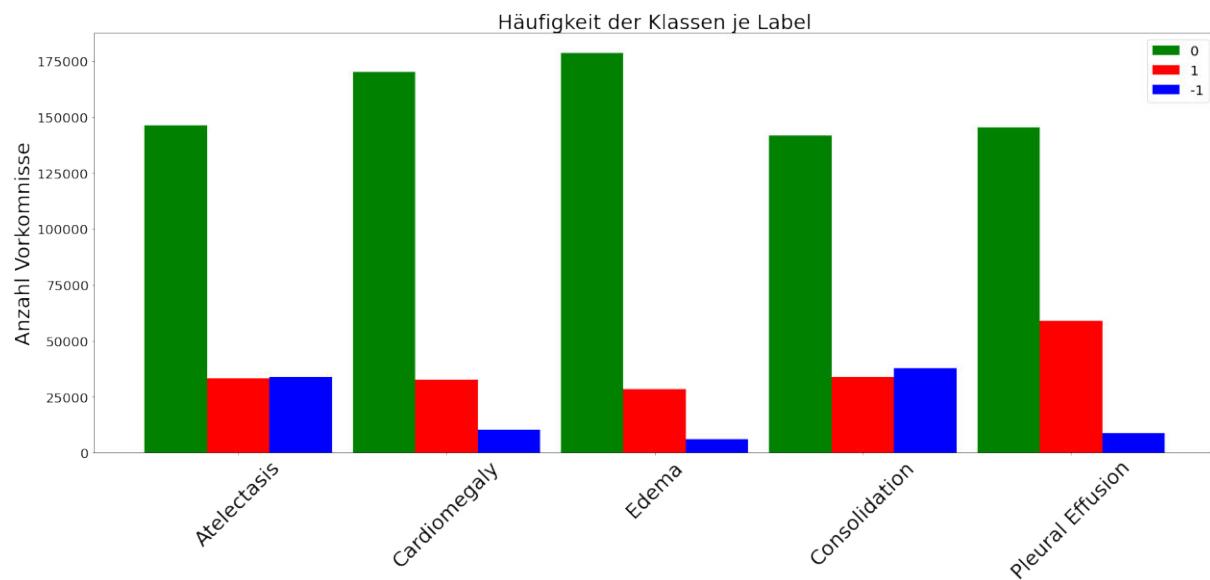


Abbildung 9.2: Verteilung der Klassen je Label aller Label nach dem Ausbalancieren der Vergleichslabel. Der Datensatz weist insgesamt weiterhin starke Imbalance auf.

Label	Trainingsdaten (Klassen: 1 und -1)	Des. Validie- rungsdaten (Klassen: 1)	Differenz	Maßnahme
Atelectasis	30.0%	29.0%	+ 1%	Keine
Cardiomegaly	15.7%	19.2%	- 3.5%	Oversampling
Edema	29.2%	14.1%	+ 15.1%	Undersampling
Consolidation	19.0%	34.2%	- 15.2%	Oversampling
Pleural Effusion	43.8%	28.6%	+ 15.3%	Undersampling

**Tabelle 9.2:** Die Ausgangssituation und daraus resultierende Maßnahme für das Angleichen der Anteile der Vergleichslabel des Trainingsdatensatz an die des designierten Validierungsdatensatzes.



**Abbildung 9.3:** Verteilung der Klassen je Label für die fünf Vergleichslabel nach dem Angleichen der Trainingsdaten an die designierten Validierungsdaten.

aussortiert, um das Undersampling zu vervollständigen. Bei Cardiomegaly und Consolidation wurden die entsprechenden Bilder dem Datensatz mehrfach hinzugefügt, sodass die Zielverteilung erreicht wird. Insgesamt enthält der so angepasste Datensatz etwa 213.000 Bilder und somit etwas weniger als der Originaldatensatz. Das Ergebnis wird in Abbildung 9.3 veranschaulicht.

### 9.1.3 Template Matching

Sowohl für die frontale als auch für die laterale Ansicht wurde ein Template erstellt. Mittels diesen wurden mit dem in Abbildung 8.2 dargestellten Template Matching Algorithmus `TM_SQDIFF_NORMED` für jedes Bild die Eckkoordinaten der erkannten Bounding-Box in der CSV-Datei festgehalten. Der Dataset-Klasse wurde ein Parameter zum Ausschneiden dieses Bildbereiches hinzugefügt.

### 9.1.4 Trainingsloop

Die Funktion `utils.basic_train_model()` beschreibt einen Standard-Trainingsloop, der auf der Implementierung aus [117] beruht und um mehrere Features und Steuerparameter erweitert wurde. Dabei wird, wenn möglich, das *Automatic Mixed Precision* Paket `torch.cuda.amp` [118] eingesetzt, das das Training um etwa 20% beschleunigte. Dieser Geschwindigkeitsvorteil entsteht, da für manche Rechenoperationen der Datentyp `torch.float16` ausreichend genau und deutlich schneller ist als der üblich genutzte `torch.float32`. Dies wird durch das Paket an den möglichen Stellen ausgenutzt [118].

### 9.1.5 Patientensensitive Datenaufteilung

Das Aufteilen der Trainingsdaten in Trainings- und Validierungsdaten geschieht unter Berücksichtigung der Patienten ID. Da der Datensatz aus rund 220.000 Bildern von etwa 65.000 Patienten besteht, ist offensichtlich, dass von einigen Patienten mehrere Aufnahmen durchgeführt wurden. Würden nun die Bilder des gleichen Patienten auf unterschiedliche Teildatensätze gesplittet, gäbe es ein gewisses Informationsleck zwischen den Trainings- und Validierungsdaten, weswegen die Patienten ID beim Split berücksichtigt werden sollte. Die Implementierung nutzt dafür die Klasse `GroupShuffleSplit` aus `sklearn.model_selection`.

## 9.2 Grundlage

Um die Effekte der weiteren Experimente beurteilen zu können, muss zunächst ein grundlegender Vergleichswert geschaffen werden. Dafür wurde eine 5-fache Kreuzvalidierung (KV) durchgeführt mit je 80% Trainings- und 20% Validierungsdaten. Im Folgenden werden die auf den so entstandenen Validierungsdaten erreichten AUC-Werte, falls eine k-fache Kreuzvalidierung vorgenommen wurde, in der Form  $\bar{x} \pm s$  als gemittelter AUC über die fünf Vergleichslabel angegeben. Es handelt sich ausdrücklich nicht um die designierten Validierungsdaten des CheXpert-Datensatzes.

Die grundlegende Trainingskonfiguration sah wie folgt aus. Mit einer Batchgröße von 32 wurden die Modelle für bis zu 15 Epochen mit dem Adam-Optimisierer unter Verwendung eines WeightDecays von 1e-4 mit einer initialen Lernrate von 1e-3 trainiert. Bei einem Plateau wurde die Lernrate um den Faktor 0.5 reduziert. Außerdem fand nach 10 Epochen eine Reduzierung um den Faktor 0.1 statt. Wurde über fünf Epochen keine Verbesserung des Losses erreicht, wurde das Training frühzeitig beendet („Early-Stopping“). Als Verlustfunktion kam der mit den positiven Klassen gewichtete Binary Cross Entropy Loss zum Einsatz. Für das Training wurden die Bilder zunächst auf die Auflösung  $256 \times 256$  reskaliert und dann zentriert ein Bereich der Größe  $224 \times 224$  ausgeschnitten. Zusätzlich wurde `torchvision.transforms.RandomHorizontalFlip()` angewendet. Die Validierungsdaten wurden direkt auf  $224 \times 224$  reskaliert. Normalisiert wurden die Bilder mit dem empirischen Mittelwert und Standardabweichung der ImageNet-Bilder. Für die uncertain Label wurde die U-Ones Strategie angewendet. Das Modell wurde nur in seiner besten Ausprägung nach dem Loss gespeichert. Damit erreichte ein DenseNet-121 einen mittleren AUC von  $0.861 \pm 0.009$ .

Dies wirft allerdings eine grundlegende Frage der Herangehensweise auf.

Da die Gütebeurteilung der Modelle ausschließlich anhand der fünf Vergleichslabel erfolgt, erscheint es zweckmäßig, das Modell auch nur für diese fünf Label zu trainieren. Dies ist zunächst kontraintuitiv, da so neun der 14 Label für das Training ignoriert werden. Tatsächlich wäre es sinnvoller, ein Modell, das in der Praxis eingesetzt werden soll, auf allen Labels zu trainieren. Die Beschränkung auf fünf Label ist ausschließlich daher motiviert, da die Evaluation aufgrund der limitierten Validierungsdaten ausschließlich anhand dieser Pathologien passiert.

In der Literatur lassen sich beide Ansätze finden. Während beispielsweise Pham et al. [95] explizit 14 Ausgabeneuronen verwenden, werden von Yuan et al. [102] nur die fünf Vergleichslabel genutzt, und genauer gesagt jeweils ein binärer Klassifizierer trainiert. Beide Methoden erreichten sehr ähnliche Ergebnisse.

Um den Effekt zu beurteilen wurde daher ein Experiment durchgeführt. Dabei verbesserte sich der AUC-Wert des DenseNet-121 leicht auf  $0.864 \pm 0.005$ . Aus diesem Grund werden in den folgenden Experimenten, sofern nicht anders angemerkt, die Modelle nur auf den Vergleichslabeln trainiert und auch nur diese vorhergesagt.

Neben dem DenseNet-121 wurde um die Effekte der Erweiterungen zu beurteilen, auch ein MobileNet V2 [119] verwendet. Dieses erzielte mit einer Epochenzahl von 20 mit Training auf den Vergleichslabeln einen AUC von  $0.869 \pm 0.003$ .

Des Weiteren wurde eine 5-fache Kreuzvalidierung mit einem SqueezeNet [120] und einer ResNext-Architektur [121] durchgeführt, um das DenseNet-121 und MobileNet per se zu beurteilen. Das SqueezeNet erreichte einen mittleren AUC von  $0.824 \pm 0.008$  und das ResNext-50 ( $32 \times 4d$ )  $0.868 \pm 0.002$ . Dies zeigt, dass sowohl das DenseNet-121 als auch das MobileNet V2 als Architekturen geeignet sind. Das ResNext-50 kann als Kandidat für späteres Ensembling identifiziert werden.

### 9.3 Erweiterungen

Zunächst wurden die Effekte der Anpassungen einzeln betrachtet, um festzustellen, ob sie zu einer Verbesserung führen. Ausgehend von der Grundlagen-Implementierung wurden die Adaptionen eingefügt und über eine k-fache Kreuzvalidierung deren Auswirkung bemessen.

Untersucht wurden dabei die folgenden Maßnahmen, deren Resultate in Tabelle 9.3 aufgeführt sind.

- M1** Bilddatumentierung: CLAHE.
- M2** Bilddatumentierung: Erweiterte online Augmentierung mit RandomRotation, RandomAffine und RandomErasing.
- M3** Bilddatumentierung: Erweiterte online Augmentierung mit RandomRotation, RandomAffine ohne RandomErasing.
- M4** Modell-Architektur: Hinzufügen von Dropout-Regularisierung.
- M5** Datensatz: Ausgleichen der Vergleichslabel.

- M6** Datensatz: Anpassen der Vergleichslabel-Verteilung an die designierten Validierungsdaten.
- M7** Datensatz: U-Ones+LSR statt U-Ones.
- M8** Datensatz: U-Mixed statt U-Ones.
- M9** Datensatz: Nur frontale Bilder verwenden.
- M10** Eingabebild: Template Matching.
- M11** Eingabebild: Höhere Auflösung von  $320 \times 320$ .
- M12** Verlustfunktion: Focal-Loss.
- M13** Verlustfunktion: LSEP-Loss (Smooth PWE-Loss).

Die Ideen zu den beiden Maßnahmen **M8** und **M9** stammen aus der Implementierung [104] zum DAM-Loss von Yuan et al. [102]. Mit U-Mixed soll dabei bezeichnet werden, dass je nach Label U-Ones bzw. U-Zeros angewendet wird. Yuan et al. [104] nutzen für Edema und Atelectasis U-Ones und für Cardiomegaly, Consolidation und Pleural Effusion U-Zeros. Allerdings liefern die Autoren keine Begründung für dieses Vorgehen und diese geht auch nicht offensichtlich aus den Verteilungen der Daten hervor. Vermutlich orientierten sie sich dabei an den Resultaten von Irvin et al. [17], bei dem die verschiedenen Ansätze verglichen wurden. Dort erzielte für Atelectasis, Edema und Pleural Effusion U-Ones bessere Ergebnisse als U-Zeros, während für Cardiomegaly und Consolidation U-Zeros zu einem höheren AUC-Wert führte.

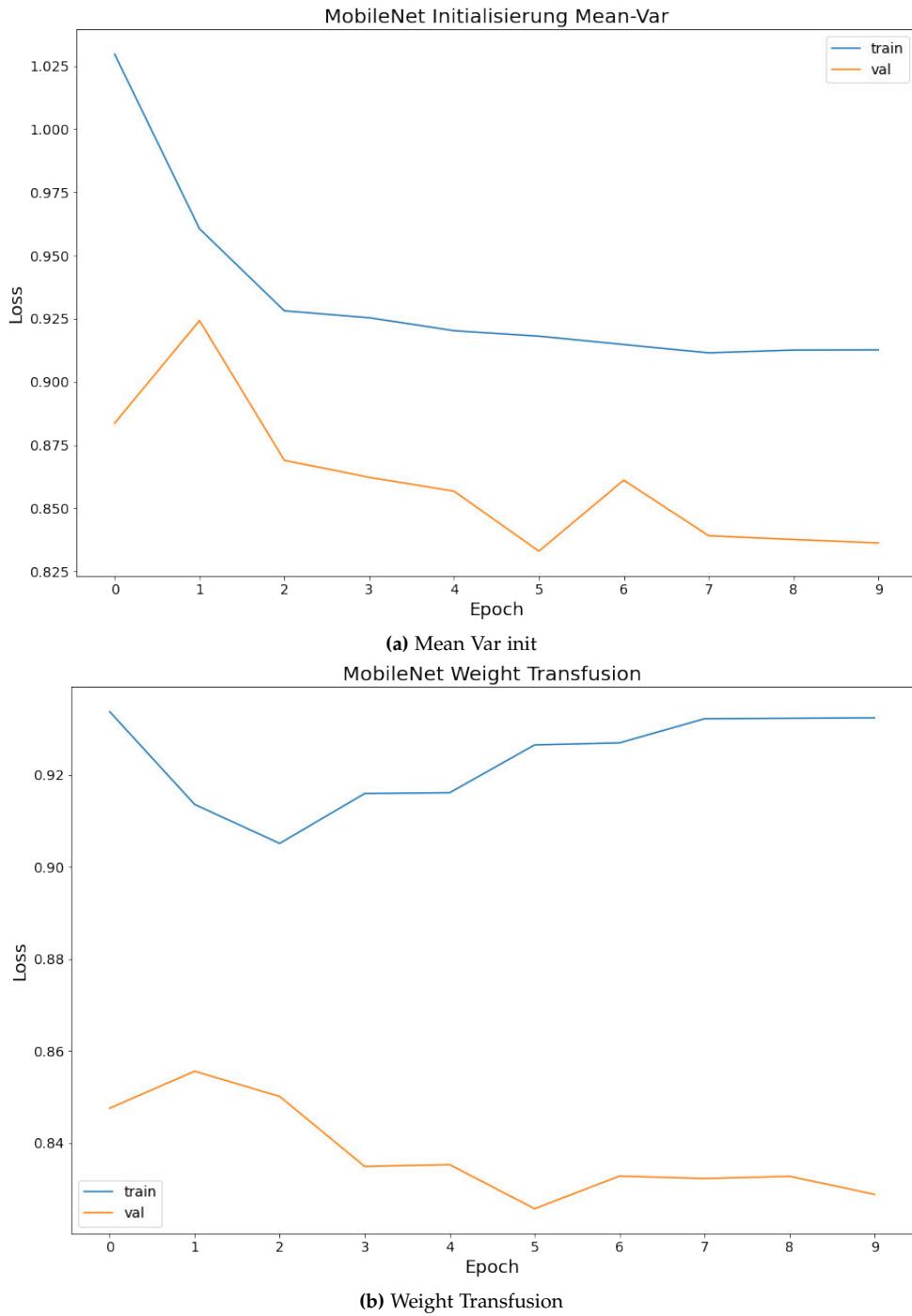
Die Resultate in Tabelle 9.3 zeigen einerseits, dass keine der Maßnahmen zu besonders relevanten Verbesserungen geführt hat. Andererseits zeigen sie auch, dass die Unterschiede zwischen den KV-Folds ebenfalls sehr gering sind. Positiv hervorzuheben ist einzig die stärkere Bildaugmentierung ohne RandomErasing **M3**.

## 9.4 Gewichtsinitialisierung

In Anlehnung an [56] wurde mit einem MobileNet-V2 zudem das *Mean Var init* sowie die *weight transfusion* getestet. In Abbildung 9.4 sind die Werte der Verlustfunktion im Trainingsverlauf für beide Methoden dargestellt. Wie zu erwarten konvergiert die *weight transfusion* Methode schneller. Sie erreichte auch eine höheren AUC auf den Validierungsdaten (0.886) als die *Mean Var init* (0.881), wobei in beiden Fällen alle 200 Batches eine Intra-Epoch Evaluation angewendet wurde. Hier ist bemerkenswert, dass mit der *weight transfusion* der beste Validierungsdaten-AUC in der ersten Epoche erreicht wurde, während die bei *Mean Var init* erst in der letzten Epoche geschah. Allerdings sollte berücksichtigt werden, dass auch das Standard Transfer Learning Verfahren mit den ImageNet Gewichten zu vergleichbaren Resultaten führt (vgl. Abbildung B.16).

<b>Maßnahme</b>	<b>Modell</b>	<b>AUC</b>
M1	DenseNet-121	$0.868 \pm 0.004$
M2	MobileNet V2	$0.869 \pm 0.004$
M3	MobileNet V2	$0.874 \pm 0.003$
M4 / M4 mit 14 Labeln	DenseNet-121	$0.847 \pm 0.006$ / $0.847 \pm 0.006$
M5	DenseNet-121	$0.847 \pm 0.004$
M6	DenseNet-121	$0.860 \pm 0.006$
M7	MobileNet V2	$0.871 \pm 0.007$
M8	MobileNet V2	$0.870 \pm 0.008$ (3-fache KV)
M9	MobileNet V2	$0.867 \pm 0.009$
M10	MobileNet V2	$0.868 \pm 0.004$
M11	DenseNet-121 / DenseNet-161	$0.861 \pm 0.001$ / $0.861 \pm 0.003$ (je 2-fache KV)
M12	MobileNet V2	$0.869 \pm 0.003$
M13	MobileNet V2	$0.832 \pm 0.005$

**Tabelle 9.3:** Resultate des Vergleichs der Maßnahmen. Gezeigt wird der AUC auf den Validierungsdaten einer  $k$ -fachen Kreuzvalidierung in der Form arithmetischer Mittelwert  $\pm$  Standardabweichung. Wenn nicht anders angegeben gilt  $k = 5$ .



**Abbildung 9.4:** Verlauf der Loss-Werte für ein MobileNet V2 mit den Strategien Mean Var init und weight transfusion aus [56]. Bei der weight transfusion wurde die erste Hälfte der Parameter übernommen und der Rest zufällig initialisiert.

## 9.5 Diagnose like a radiologist

Für die AG-CNN Methode von Guan et al. [100] wurde zunächst ein Experiment durchgeführt, das das Vorgehen der Autoren repliziert. Der mittlere AUC für die fünf Vergleichsklassen ergab sich als 0.875, 0.789, 0.876 für GB, LB und FB. Durch das Hinzufügen der Intra-Epoch Evaluation konnte selbst mit nur 25 Trainingsepochen eine leichte Verbesserung auf 0.876, 0.814, 0.879 erzielt werden.

Dass der Fusion Branch nur marginal besser ist als der Global Branch, lässt darauf schließen, dass der Local Branch nur sehr geringen Einfluss hat. Betrachtet man die Gewichtungen, die der Fusion Branch für die Eingaben aus dem GB und LB erlernt hat, wird klar, dass der lokale Branch deutlich weniger Einfluss auf das Endergebnis hat als der GB. Die Absolutbeträge der Gewichte betragen als  $\bar{x} \pm s$  für den GB  $0.0119 \pm 0.0520$  und für den LB  $0.0037 \pm 0.0185$ .

In Tabelle 9.4 findet sich eine Aufschlüsselung der Gewichte nach den Pathologien. Daraus geht hervor, dass je nach Label der Local Branch unterschiedlich stark berücksichtigt wird. So ist dieser bei Cardiomegaly am geringsten und bei Consolidation am höchsten. Mit dem Hintergrundwissen zu den Pathologien, die in Abschnitt 5.4.1 beschrieben werden, wäre ein Interpretationsansatz, dass die Bildausschnitte groß genug sind, um die Verschattungen der Alveolen zu erkennen, nicht aber die Vergrößerung des Herzens. Zudem wird der Bildausschnitt auf die ursprüngliche Bildauflösung vergrößert. Wird also korrekterweise das Herz als Läsionsbereich erkannt und aus dem Bild ausgeschnitten, wird es anschließend künstlich vergrößert, was offensichtlich die Erkennung einer krankhaften Herzmuskelvergrößerung (Cardiomegaly) beeinträchtigt. Der gleiche Versuchsaufbau wurde anschließend mit einem geringeren Threshold  $\tau = 0.55$  statt wie oben  $\tau = 0.7$  durchgeführt. Die AUC-Werte für GB, LB und FB betrugen dabei 0.874, 0.851, 0.873. Die absoluten Gewichtungen für den GB  $0.0079 \pm 0.0343$  und den LB  $0.0057 \pm 0.0260$ . Der Einfluss von  $\tau$  stellt sich also wie erwartet und analog zu [100] dergestalt dar, dass ein niedrigeres  $\tau$  zwar zu höheren AUC-Werten des LB führt, das Gesamtergebnis aber nicht verbessert wird. Die Betrachtung der Gewichte zeigt zudem, dass in diesem Fall das Verhältnis zwischen den Branches ausgeglichener ist.

<b>Label</b>	<b>GB <math>\bar{x} \pm s</math></b>	<b>LB <math>\bar{x} \pm s</math></b>	<b>Quotient GB / LB</b>
Atelectasis	$0.0401 \pm 0.0095$	$0.0152 \pm 0.0033$	$2.6321 \pm 2.8350$
Cardiomegaly	$0.0591 \pm 0.0133$	$0.0173 \pm 0.0033$	$3.4086 \pm 4.0118$
Edema	$0.0602 \pm 0.0150$	$0.0225 \pm 0.0047$	$2.6803 \pm 3.1863$
Consolidation	$0.0319 \pm 0.0077$	$0.0153 \pm 0.0032$	$2.0792 \pm 2.3671$
Pleural Effusion	$0.0612 \pm 0.0141$	$0.0211 \pm 0.0040$	$2.9000 \pm 3.5588$

**Tabelle 9.4:** Vergleich der absoluten Gewichtungen des Fusion Branches für Global und Local Branch, aufgeschlüsselt nach den fünf Vergleichslabellen.

## 9.6 Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels

Zunächst wurde diese Methode so implementiert, wie sie von Pham et al. [95] beschrieben wird. Allerdings war es damit nicht möglich, die Resultate (in einem Toleranzbereich) zu reproduzieren. Auffällig war dabei, dass der beste AUC eines einzelnen Modelles, der in [95] als 0.894 mit einem DenseNet-121 berichtet wird, zwar mit der Implementierung der vorliegenden Arbeit erreichbar ist, und vereinzelt auch übertragen wurde. Bei dem Effekt des Ensemblings, das lediglich aus der Mittelwertbildung besteht, aber sehr große Abweichungen zwischen [95] und den vorliegenden Experimenten existieren. Bei Pham et al. [95] erreichte das Ensemble einen AUC von 0.94, die Verbesserung zum besten Einzelmodell betrug also 0.046. In den hier durchgeföhrten Experimenten war die Verbesserung deutlich geringer (siehe Tabelle 9.6). In Rücksprache mit den Autoren von [95] wurde die Implementierung der vorliegenden Arbeit bestätigt. Daher ist es unklar, woher diese Diskrepanz röhrt.

Um die Resultate zu optimieren wurden einige Experimente durchgeföhr, deren Ergebnisse in Tabelle 9.5 aufgeführt sind. Dabei wurden Strategien zur Modifikation der verwendeten Daten, eine andere LSR (LSR-V2), sowie Anpassungen des Trainingsablaufs bezüglich dem Einfrieren von Schichten und der Reduzierung der Lernrate erprobt. Als Modell wurde ein DenseNet-121 verwendet. Außerdem wurde die Intra-Epoch Evaluation mit  $N = 400$  angewendet.

Die zusätzlich getestete LSR-V2 besteht darin, für die Label Cardiomegaly und Consolidation die U-Zeros+LSR mit den Grenzwerten 0 und 0.3 und für Atelectasis, Edema und Pleural Effusion U-Ones+LSR mit den Grenzwerten 0.55 und 0.85 anzuwenden. Dieses Vorgehen liegt darin begründet, dass Irvin et al. [17] für Cardiomegaly und Consolidation das U-Zeros Vorgehen zu besseren Ergebnissen führte als U-Ones und für die anderen Label das Gegenteil gilt.

Außerdem wurde die Oversampling-Methode von Yuan et al. [104] angewendet. Dabei werden die positiven Bilder der Label Cardiomegaly und Consolidation im Datensatz dupliziert. Dies wird in dieser Arbeit als *simple oversample* bezeichnet.

Mit den Erkenntnissen aus den Experimenten, die in Tabelle 9.5 beschrieben werden, wurden erneut Trainingsläufe durchgeföhr um die bestmöglichen Resultate zu erreichen. Als Eingabegröße wurde  $320 \times 320$  verwendet, die durch zufälliges Ausschneiden aus den zuvor auf  $380 \times 380$  skalierten Bildern, erzeugt wurde. Außerdem wurde zufälliges horizontales Spiegeln sowie `transforms.RandomAffine(degrees=(-5, 5), translate=(0.01, 0.15), scale=(0.9, 1.2))` und `transforms.RandomRotation(degrees=5)` angewendet. Die Normalisierung der Bilder geschah mit dem empirischen Mittelwert und der Standardabweichung des ImageNet-Datensatzes. Die uncertain Label wurden mit der U-Ones+LSR Methode aus [95] mit den dort genutzten Grenzwerten 0.55 und 0.85 behandelt. Um diverse Einzelmodelle zu erhalten wurde das Ersetzen der Label für jedes Modell neu durchgeföhr, sodass in jedem Trainingslauf leicht unterschiedliche Zielwerte vorgegeben werden. Für das CT wurde der Adapted Dataset verwendet und für das FT der normale Datensatz, erweitert durch das simple

Maßnahme	AUC
Mit simple oversample, ohne Freeze	CT: 0.7851, FT: 0.8808
Ohne simple oversample, ohne Freeze	FT: 0.8721
Ohne simple oversample, mit Freeze	FT: 0.8621
Mit simple oversample, mit Freeze, mit LSR-V2	FT: 0.8521
Mit simple oversample, mit Freeze, mit LSR-V2, ohne OoD-2.5, ohne Bilder bei denen alle Label negativ sind	FT: 0.8365
Mit simple oversample, mit Freeze, mit LSR-V2, ohne OoD-3.5, mit Bildern bei denen alle Label negativ sind	FT: 0.8440
Adapted Dataset, mit Freeze	CT: 0.8103, FT: 0.8496
Adapted Dataset, ohne Freeze	CT: 0.8073, FT: 0.8742
Adapted Dataset für CT dann normalen Datensatz, mit simple oversample, mit Freeze	CT 0.8287, FT: 0.8597
Adapted Dataset für CT dann normalen Datensatz, mit simple oversample, ohne Freeze	CT 0.8287, FT: 0.8732
Adapted Dataset, ohne Freeze, ReduceLROnPlateau statt nach Epoche	CT: 0.8204, FT: 0.8660
Adapted Dataset für CT dann normalen Datensatz, mit simple oversample, mit Freeze, ReduceLROnPlateau statt nach Epoche	CT 0.8289, FT: 0.8631
Adapted Dataset für CT dann normalen Datensatz, mit simple oversample, ohne Freeze, ReduceLROnPlateau statt nach Epoche	CT 0.8289, FT: 0.8886
Adapted Dataset für CT dann normalen Datensatz, mit simple oversample, ohne Freeze, ReduceLROnPlateau statt nach Epoche, Normalisierung mit CheXpert-Lageparametern statt ImageNet	CT 0.8287, FT: 0.8835

**Tabelle 9.5:** Resultate der Experimente zur Optimierung der Methode aus[95]. An relevanten Stellen wird neben dem FT-Wert auch der CT-Wert angegeben. Mit ReduceLROnPlateau wurde die Lernrate mit dem Faktor 0.1 multipliziert, wenn sich der AUC-Wert nicht zur vorherigen Epoche verändert hat. **CT:** Conditional Training, **FT:** Fine-tuning, **Freeze:** Einfrieren aller Schichten außer der Letzten.

Modell	Mit Freeze			Ohne Freeze		
	ohne Bayes, ohne TTA	ohne Bayes, mit TTA	mit Bayes, mit TTA	ohne Bayes, ohne TTA	ohne Bayes, mit TTA	mit Bayes, mit TTA
DenseNet-121	0.870	0.873	0.869	0.884	<b>0.892</b>	<b>0.892</b>
DenseNet-169	0.875	0.875	0.865	<b>0.895</b>	0.890	0.887
DenseNet-201	0.884	0.885	0.878	<b>0.893</b>	0.889	0.865
Xception	0.875	0.875	0.869	<b>0.892</b>	0.887	0.885
Inception	0.874	0.882	0.883	0.891	<b>0.900</b>	0.897
Ensemble	0.894	0.892	0.886	<b>0.903</b>	0.902	0.896

**Tabelle 9.6:** Resultate der Experimente zu [13]. Verglichen werden die AUC-Werte der Einzelmodelle sowie des Ensembles auf den Validierungsdaten. Dabei werden Konfigurationen mit und ohne Anwendung der Bayes'schen Regel und der TTA betrachtet. Dies geschieht jeweils für einen Versuch mit dem Einfrieren aller außer der letzten Schicht nach dem Conditional Training, und einen ohne diesen Freeze. Die beste AUC je Modell ist fett gedruckt.

Oversampling. Aufgrund der Hardwarbeschränkungen konnte nur mit einer Batchsize von 12 trainiert werden. Wie in [95] beschrieben wurde sowohl das CT als auch das FT für fünf Epochen durchgeführt und auf allen 14 Labeln trainiert. Statt nach jeder Epoche wurde die Lernrate bei Plateaus reduziert. Die Intra-Epoch Evaluation wurde alle 100 Batches durchgeführt. Mit dieser Trainingskonfiguration wurde ein Versuch mit dem Einfrieren der Schichten zwischen den Trainingsphasen CT und FT und ein Lauf ohne diesen Freeze durchgeführt. Die Resultate werden in Tabelle 9.6 aufgeführt. Daraus wird ersichtlich, dass sich das Ensembling generell positiv ausgewirkt hat, während die Anwendung der Bayes'schen Regel die AUC-Werte eher senkte. Der Einfluss der TTA kann als neutral bis leicht positiv beschrieben werden. Das Einfrieren der Schichten führte insgesamt zu schlechteren Ergebnissen, jedoch ist der Effekt des Ensemblelings dafür größer.

Mit diesem Freeze wurden zwei weitere Experimente durchgeführt. Zum einen wurde die Epochenzahl auf zehn erhöht. Für den zweiten Versuch wurde außerdem die Intra-Epoch Evaluation deaktiviert. Diese Resultate befinden sich in Tabelle 9.7. Sie zeigen auf, dass eine höhere Epochenzahl als Pham et al. [13] angegeben zu Verbesserungen sowohl bei den einzelnen Modellen als auch beim Ensembling geführt hat. Dennoch bleiben die Ergebnisse noch leicht hinter denen ohne Einfrieren zurück. Außerdem zeigt sich erneut der bedeutende Effekt der Intra-Epoch Evaluation. Beim Ensembling ist ohne diese der AUC-Wert etwa um 0.015 – 0.02 geringer und bei den Einzelmodellen beträgt die Differenz sogar bis zu 0.03.

Modell	Mit Freeze, 10 Epochen			Mit Freeze, 10 Epochen, keine Intra-Epoch Eva- luation		
	ohne Bayes, ohne TTA	ohne Bayes, mit TTA	mit Bayes, mit TTA	ohne Bayes, ohne TTA	ohne Bayes, mit TTA	mit Bayes, mit TTA
DenseNet-121	<b>0.886</b>	<b>0.886</b>	0.880	0.856	0.865	0.860
DenseNet-169	0.889	<b>0.890</b>	0.889	0.860	0.868	0.869
DenseNet-201	<b>0.888</b>	0.886	0.879	0.855	0.860	0.866
Xception	0.888	0.889	<b>0.890</b>	0.856	0.865	0.862
Inception	0.884	0.887	<b>0.888</b>	0.856	0.865	0.867
Ensemble	<b>0.899</b>	0.896	0.893	0.876	0.879	0.878

**Tabelle 9.7:** Resultate der weiteren Experimente zu [13]. Hierbei wurden alle außer der letzten Schicht nach dem Conditional Training eingefroren und für zehn Epochen trainiert. Außerdem wurde ein Versuch ohne Intra-Epoch Evaluation durchgeführt. Die beste AUC je Modell ist fett gedruckt.

## 9.7 CNN als Feature-Extractor

Für die Methode von Allaouzi et al. [52] bei der ein CNN als Feature-Extractor und eine Logistische Regression als Klassifikator genutzt wird, wurde ein DenseNet-121 verwendet, das mit einer Bildauflösung von  $512 \times 512$  trainiert wurde und einen AUC von 0.904 erreichte. Dessen letzte Schicht wurde durch eine Identity-Schicht ersetzt, die  $f(x) = x$  abbildet. Somit besteht die Ausgabe des CNNs für  $N$  Datenpunkte aus einer Feature-Repräsentation der Dimension  $N \times 1024$ , die als Eingabe für andere Klassifizierer verwendet werden kann. Mit diesem CNN wurde ein Forward-Pass über alle Bilder ausgeführt und die Ausgaben gespeichert. Mittels der Bibliothek `skmultilearn` [111] wurden die beiden Problemtransformationsmethoden Binary Relevance (BR) und Classifier Chain (CC) für verschiedene Basis-Klassifizierer implementiert. Die LP-Methode wurde nicht angewandt, da die Berechnungszeit deutlich länger ist und zu erwarten ist, dass die Ergebnisse nicht besser wären (vgl. [52]). Ferner wurde der Effekt der PCA, wie sie in Kapitel 8.5 beschrieben wird, untersucht. Die Ergebnisse der Experimente befinden sich in Tabelle 9.8.

Daran ist zu erkennen, dass zwar keine der Konfigurationen besser ist, als das ursprüngliche CNN, die AUC-Werte aber durchaus auf dem Niveau von CNNs liegen. Ein solches Modell könnte zudem als Ensemble-Mitglied sinnvoll sein, da es die Diversität erhöht. Die besten Resultate erreichte das AdaBoost-Verfahren, das intern eine Logistische Regression verwendet.

Zwischen den AUC-Werten von BR und CC sind nur sehr geringe Unterschiede fest-

Basis-Klassifizierer	Anzahl-Trainingsdaten	Mittlerer AUC (BR / CC)	Mittlerer AUC mit PCA(n=50) (BR / CC)	Mittlerer AUC mit PCA(n=36) (BR / CC)
AB(n_estimators=25)	10.000	0.858 / 0.857	0.870 / 0.870	0.863 / 0.865
LR(max_iter=10000)	10.000	0.874 / 0.871	0.876 / 0.871	0.884 / 0.880
AB( LR(max_iter=1000))	10.000	0.886 / 0.882	0.890 / 0.874	0.883 / 0.881
LR(max_iter=10000)	N	0.886 / 0.881	0.889 / 0.886	0.890 / 0.888
AB( LR(max_iter=1000))	N	0.891 / 0.887	0.889 / 0.888	0.890 / 0.888
AB( LR(max_iter=1000), n_estimators=100)	N	0.889 / n.d.	0.889 / n.d.	0.890 / n.d.
RFC()	N	0.858 / n.d.	0.875 / n.d.	0.875 / n.d.
AB( RFC( n_estimators=50, max_depth=15), n_estimators=25)	N	0.860 / n.d.	0.831 / n.d.	0.832 / n.d.

**Tabelle 9.8:** Resultate der Experimente zu [52] in Form der AUC auf den CheXpert-Validierungsdaten. Für die Basis-Klassifizierer wurden die Implementierungen aus sklearn genutzt. Bei den Trainingsdaten entspricht N dem gesamten CheXpert-Trainingsdatensatz, bei dem das simple Oversampling (vgl. Abschnitt 9.6) angewendet wurde, wodurch N = 265.197. Die PCA beinhaltet die 50 bzw. 34 ersten Hauptkomponenten. **Abkürzungen:** AB = AdaBoostClassifier, LR = LogisticRegression, RFC = RandomForestClassifier, n.d. = nicht durchgeführt.

stellbar. Daher wurden die Konfigurationen, die besonders zeitaufwendig sind, nur mit der BR-Transformation durchgeführt. Zudem ist der Effekt der PCA bemerkenswert. Dank der deutlich geringeren Dimensionalität wird nicht nur das Fitting beschleunigt, auch die Genauigkeit der Klassifizierer wurde – mit Ausnahme der letzten Konfiguration, bestehend aus der Kombination von AdaBoost und RandomForest – in fast allen Fällen leicht verbessert. Auch hier verhalten sich BR und CC sehr ähnlich. Einzig bei der Verwendung des AdaBoostClassifiers mit einer Logistischen Regression und 10.000 Trainingsdaten ist eine größere Diskrepanz festzustellen. Besonders auffällig ist, dass in dieser Situation die BR-Transformation mit 50 Hauptkomponenten bessere Resultate erzielte als mit 36, während bei der CC-Transformation das gegenteilige Verhalten zu beobachten war.

## 9.8 Snapshot Ensemble

Um das Snapshot Ensemble zu testen, wurde die MobileNet V2 Architektur [119] gewählt. Wie in Kapitel 2.5 beschrieben, unterscheidet sich das Snapshotting durch die Verwendung einer zyklischen Lernrate vom horizontalen Ensembling. Die Grundlegende Trainingskonfiguration sah wie folgt aus.

Es wurde ein Ensemble aus vier Modellen gebildet. Alle zehn Epochen (im Folgenden als „Cooldown“ bezeichnet) wurde dafür ein Snapshot gespeichert. Nachdem die letzte Epoche trainiert wurde, wurde ebenfalls ein Modell-Checkpoint erzeugt. Insgesamt wurde für 39 Epochen mit einer Batchgröße von 64 trainiert. Als Verlustfunktion kam der BCE Loss zum Einsatz, bei dem die positiven Datenpunkte nach ihrer Auftrittswahrscheinlichkeit gewichtet wurden. Als Optimisierer wurde der Stochastic-Gradient-Descent verwendet mit einem Weight-Decay von 1e-4. Das Modell wurde nur für die fünf Vergleichslabel trainiert und die Intra-Epoch Evaluation kam zum Einsatz um die besten Modell-Ausprägungen zu ermitteln. Die Bildauflösung betrug  $320 \times 320$  und als Augmentierung kam `RandomHorizontalFlip()` sowie `RandomAffine(degrees=(-5, 5), translate=(0.01, 0.15), scale=(0.9, 1.2))` und `RandomRotation(degrees=5)` aus der `torchvision` Bibliothek zum Einsatz. Normalisiert wurden die Eingaben mit den ImageNet Lageparametern. Die uncertain Label wurden mit der U-Ones+LSR Methode aus [95] behandelt. Die zyklische Lernrate verlief innerhalb der Grenzen 1e-4 und 1e-2 und die Schrittgröße wurde so berechnet, dass ein Zyklus (a) die gesamten Trainingsepochen oder (b) eine Cooldown-Phase umfasst. Der zyklische Lernraten-Scheduler wurde demnach definiert, indem eine zyklische Schrittgröße als

$$\frac{N}{2} \left\lceil \frac{|\mathcal{D}|}{b} \right\rceil \quad (9.1)$$

berechnet wurde, die sowohl für die Steigerung als auch die Reduzierung der Lernrate genutzt wurde. Dabei ist  $b$  die Batchsize,  $|\mathcal{D}|$  die Länge des Datensatzes und  $N$  entweder (a) die Gesamtanzahl der Epochen oder textit(b) der Cooldown-Wert.

Um die Diversität der Modelle weiter zu erhöhen, wurde zudem untersucht, wie sich das Zurücksetzen der Gewichte einzelner Schichten auswirkt. Um den Trainingserfolg dabei nicht gänzlich zu verwerfen, werden die Gewichte normalverteilt mit dem Mittelwert und der Standardabweichung der erlernten Parameter initialisiert. Dies ist durch die *weight transfusion* inspiriert (vgl. Abschnitt 4.3).

Dies hat weiterhin den Vorteil, dass die neuen Parameterwerte zumindest in der Größenordnung mit den vorherigen übereinstimmen. Ansonsten könnte es dazu kommen, dass zufälliges Reinitialisieren große Gradienten und damit Gewichtsänderungen nach sich zieht, was die früheren Feature „zerstören“ könnte, da diese nicht eingefroren werden (vgl. Fine-tuning [122]). Dieser Effekt sollte durch die Initialisierung mit dem arithm. Mittelwert  $\bar{x}$  und empirischer Standardabweichung  $s$  abgeschwächt werden, wenngleich die negativ beeinflussten Feature im weiteren Trainingsverlauf auch (wieder-)erlernt werden können.

Zusätzlich wird dies nur mit einer Wahrscheinlichkeit von  $p = 0.25$  durchgeführt, ansonsten bleiben die erlernten Gewichte erhalten. Dieses Vorgehen wurde weiterhin unterteilt.  $p$  ist demnach ein wichtiger Hyperparameter, da über ihn die Menge an

Informationsverlust gesteuert werden kann. Je größer  $p$ , desto mehr Parameter werden zurückgesetzt und desto größer ist das Potenzial, dass unterschiedliche lokale Minima gefunden werden, die für das Ensembling förderlich sind. Jedoch steigt damit auch der Verlust der erlernten Parameterwerte und bisher Erlerntes wird mit höherer Wahrscheinlichkeit verworfen, wodurch längere Cooldown-Phasen nötig sein könnten. In einer Konfiguration werden die ersten 100 von 158 Parametern nicht berücksichtigt, sie blieben also unverändert. In einer anderen Implementierung wurde die Reinitialisierung der Gewichte auf alle Schichten (mit  $p = 0.25$ ) angewendet.

Die Resultate dieses Experiments befinden sich in der Tabelle 9.9.

Diese zeigen, dass die Einzelmodelle durchschnittlich am besten performten, als die Reinitialisierung erst ab dem 100. Parameter stattfand und die Lernrate über den gesamten Trainingsverlauf einen Zyklus durchläuft (Strategie (a)). Der beste AUC wurde von dem Ensemble ebendieser Trainingskonfiguration erzielt. Insgesamt erzielten die Ensembles wie auch die Einzelmodelle bessere AUCs, wenn die Gewichtungen reinitialisiert wurden.

Weiterhin ist ersichtlich, dass die Einzelmodelle zwar etwas besser performen, wenn die Gewichte der früheren Schichten geschützt werden. Durch das Ensembling wird dieser leichte Nachteil der Reinitialisierung aller Parameter aber wieder ausgeglichen. Auch die Differenz zwischen Ensemble und Einzelmodell durchschnitt war am größten, wenn die Methode auf alle Parameter angewandt wurde.

Damit zeigt sich die hier erprobte Erweiterung des Snapshotting als Reinitialisierung mit den zuvor erlernten Lageparametern als vielversprechend.

Daher wurde noch ein Snapshotting-Experiment durchgeführt mit einem DenseNet-169 bei dem zusätzlich mittels TM ein Bildbereich der Größe  $320 \times 320$  ausgeschnitten wurde und der adaptierte Datensatz zum Einsatz kam. Für das Snapshotting wurde nur das letzte Drittel der Modellparameter berücksichtigt und diese mit einer Wahrscheinlichkeit von  $p = 0.25$  reinitialisiert. Auf diese Weise wurden mehrere Trainingsanpassungen kombiniert. Diese Modelle erreichten einzeln AUCs von 0.873, 0.882, 0.880 und 0.868 (Durchschnitt: 0.876) sowie im Ensemble 0.894. Die Differenz zwischen Ensemble und Durchschnitt beträgt also 0.018, was größer ist als alle Differenzen der Experimente mit der MobileNet-Architektur. Des Weiteren wurden die Einzelmodelle mittels des DAM Losses [102] Fine-tuned. Die Ergebnisse dazu sind in Tabelle 9.11 aufgeführt.

## 9.9 Boosted cascaded convnets

In Anlehnung an Kumar et al. [12] wurde eine Kaskade implementiert, die inklusive dem CNN sechs Etappen enthält. Als Backbone wurde ein MobileNet V2 verwendet. Die Modelle wurden für fünf Epochen mit einer Batchsize von 64 und dem Adam-Optimisierer mit einer Lernrate von 1e-3 trainiert. In einem ersten Versuch wurde dabei als Verlustfunktion der BCE Loss verwendet, bei dem die positiven Klassen entsprechend ihrer Häufigkeit gewichtet wurden, um die Imbalance auszugleichen. Die Resultate sind in Tabelle 9.10 dargestellt. Die Experimente zeigten, dass das Boosting in dieser Form zu keinen großen Verbesserungen geführt hat. Die besten Ergebnisse

Experiment	AUC Einzelmodelle (Durchschn.)	AUC Ensemble	Differenz
Keine Reinitialisierung, Lernrate: (a)	0.882 / 0.884 / 0.884 / 0.884 (0.884)	0.887	0.003
Keine Reinitialisierung, Lernrate: (b)	0.885 / 0.885 / 0.885 / 0.885 (0.885)	0.890	0.005
Alle Parameter reinit., Lernrate: (a)	0.884 / 0.886 / 0.888 / 0.890 (0.887)	0.899	0.012
Alle Parameter reinit., Lernrate: (b)	0.881 / 0.888 / 0.886 / 0.889 (0.886)	0.900	0.014
Reinitialisierung ab 100. Parameter, Lernrate: (a)	0.887 / 0.890 / 0.890 / 0.893 (0.890)	0.901	0.011
Reinitialisierung ab 100. Parameter, Lernrate: (b)	0.884 / 0.890 / 0.887 / 0.890 (0.888)	0.897	0.009

**Tabelle 9.9:** Resultate der Experimente für das Snapshot Ensemble. Die Beschreibung der Trainingskonfigurationen befindet sich im Fließtext. Die Spalte „Differenz“ beinhaltet die Differenz zwischen dem AUC des Ensembles und dem Durchschnitt der AUC der Einzelmodelle.

wurden in der zweiten und dritten Etappe erreicht. Die Ensembling-Vorhersage wurde wie von Kumar et al. [12] beschrieben als Mittelwert der einzelnen Etappen gebildet. Der PWE-Loss erwies sich als ungeeignet. Die Modelle waren nicht in der Lage, mit diesem zu lernen und die erreichten AUC-Werte kamen nicht über 0.7 hinaus.

## 9.10 Robust Deep AUC Maximization

Mittels der Bibliothek LibAUC [104] wurden mit der Methode von Yuan et al. [102] mehrere Modelle verschiedener Architekturtypen trainiert. Die AUC-Werte der Modelle werden in Tabelle 9.11 aufgelistet. Diese befinden sich alle etwa im Bereich  $0.9 \pm 0.01$ . Mittels dieser Methode war es also möglich, Klassifizierer hoher Güte zu erzeugen. Allerdings muss berücksichtigt werden, dass die Bibliothek zum Zu Beginn der Entstehung dieser Arbeit keine Multi-Label Klassifikation unterstützte. Es musste demnach für jedes Label ein eigenes Modell trainiert werden. Der Multi-Label-Modus wurde erst später hinzugefügt.

Die Trainingskonfiguration orientiert sich an den Angaben aus [102]. Es wird also grundlegend das simple oversampling angewendet und das Modell zunächst mit dem Cross-Entropy- und anschließend mit dem Deep AUC Maximization (DAM)-Loss aus [104] trainiert. Die Bildaugmentierung wurde aus [104] übernommen und besteht neben dem Skalieren aus `tfs.RandomAffine(degrees=(-15, 15), translate=(0.05,`

<b>Etappe</b>	<b>AUC</b>
CNN / FC1	0.872
FC2	0.875
FC3	0.875
FC4	0.872
FC5	0.873
FC6	0.872
Ensemble	0.874

**Tabelle 9.10:** Resultate aus den Experimenten zu [12]. Die Tabelle zeigt den AUC-Wert auf den Validierungsdaten je nach Etappe der Kaskade.

0.05), scale=(0.95, 1.05), fill=128) und der Normalisierung mit ImageNet-Werten. Das Pretraining wurde wie bei Yuan et al. [102] beschrieben mit einer Lernrate und WeightDecay von je 1e-5 ausgeführt, während für das Fine-tuning die Lernrate initial 0.1 war und während dem Training in zwei Schritten reduziert. Als Hyperparameter für den PESG-Optimisierer wurde  $\gamma = 500$  und der Abstand  $m = 1.0$  festgelegt. Die Intra-Epoch Evaluation wurde alle 100 Batches ausgeführt.

<b>Modell (Trainingsbesonderheit)</b>	<b>Bildgröße</b>	<b>Mittlere AUC</b>
ResNext-50	$224 \times 224$	0.896
Xception	$224 \times 224$	0.900
DenseNet-121 (LSR-V2)	$512 \times 512$	0.903
MobileNet V2 (LSR-V2)	$320 \times 320$	0.904
DenseNet-121	$320 \times 320$	0.905
DenseNet-201	$224 \times 224$	0.908
ResNet-50	$224 \times 224$	0.908
DenseNet-201	$256 \times 256$	0.908
MobileNet V2	$512 \times 512$	0.909
DenseNet-169	$320 \times 320$	0.911
Xception	$320 \times 320$	0.911
DenseNet-121 (Pretraining mit BCE)	$512 \times 512$	0.912
ResNet-50	$512 \times 512$	0.912
DenseNet-121 (LSR)	$512 \times 512$	0.912
DenseNet-201	$480 \times 480$	0.915
MobileNet V2 (Pretraining mit Focal-Loss)	$512 \times 512$	0.915
DenseNet-169 (LSR-V2)	$512 \times 512$	0.921
DenseNet-169	$512 \times 512$	0.923
DenseNet-169 (Snapshot, Nr. 1)	$320 \times 320$	0.917
DenseNet-169 (Snapshot, Nr. 2)	$320 \times 320$	0.910
DenseNet-169 (Snapshot, Nr. 3)	$320 \times 320$	0.913
DenseNet-169 (Snapshot, Nr. 4)	$320 \times 320$	0.909

**Tabelle 9.11:** Resultate der Experimente für Yuan et al. [102]. Gezeigt wird das verwendete Modell, die Bildauflösung und die AUC auf den designierten Validierungsdaten.

# Kapitel 10

## Ensembling der Methoden

Nachdem die Modelle mit den bereits erklärten Methoden trainiert wurden, ist für das Ensembling die Modellauswahl zu treffen und die Art der Kombination zu wählen.

### 10.1 Ensemble-Selektion

Für die Modell-Auswahl wurde das Vorgehen von Caruana et al. [38] implementiert, das in Kapitel 2.5.5 genauer beschrieben wird. Dabei werden die Modelle ausgewählt, indem ihr Beitrag zur Verbesserung der Ensemble-Performance bestimmt wird. In diesem Fall wird die Ensembling-Vorhersage als Mittelwert der Einzelvorhersagen gebildet und in jeder Iteration die beste Ensembling-Zusammensetzung gewählt. Dies wurde so implementiert, dass die Modelle mit oder ohne Zurücklegen gezogen werden können und das Ensemble auch mit einer Modellliste initialisiert werden kann.

Als Modellbibliothek wurde eine Ansammlung verschiedener Modell-Architekturen, Trainingsweisen, angewendeten Methodiken und Bildgrößen verwendet, die insgesamt etwa 50 Modelle umfasste. Für das Training wurden die generellen Erkenntnisse aus den Experimenten und Vorbereitungen angewendet. Im Allgemeinen wurden nur die fünf Vergleichslabel vorhergesagt, die erweiterte online Augmentierung (Maßnahme M3) und das simple oversample eingesetzt, Transfer Learning basierend auf den ImageNet Gewichten durchgeführt, eine (initiale) Lernrate von maximal 1e-3 verwendet und der BCE Loss über den `pos_weight` Parameter gewichtet. Außerdem wurde die Intra-Epoch Evaluation mit  $N \in \{100, 200, 400\}$  angewendet und die beste Modellausprägung anhand der ROC-AUC auf den Validierungsdaten ausgewählt. Die Modelle wurden mit einem Bezeichner versehen, die in den folgenden Grafiken verwendet und daher nachstehend erklärt werden.

### 10.2 Modellbibliothek

Bei `dense121_1`, `dense121_2`, `dense121_3` und `dense121_5` handelt es sich um DenseNet-121-Modelle, die mit einer Bildauflösung von  $512 \times 512$  trainiert wurden. Um Diversität zu erzeugen, wurden unterschiedliche Hyperparameterkonfigurationen gewählt. Für

dense121\_1 wurde mit den ImageNet Parametern normalisiert, bei den anderen beiden kamen Mittelwert und Standardabweichung des CheXpert Datensatzes zum Einsatz. Für dense121\_1 wurde U-Mixed genutzt, für dense121\_3 LSR-V2 und für dense121\_2 sowie dense121\_5 U-Ones+LSR. dense121\_1 wurde mit dem Stochastic Gradient Descent mit einer Lernrate von 1e-3 trainiert, für die anderen beiden wurde der Adam Optimisierer verwendet mit einer Lernrate von 1e-4 für dense121\_1 und einer Lernrate von 1e-2, die während dem Training reduziert wurde, für dense121\_2. Auch dense121\_4 wurde ähnlich trainiert, allerdings wurde kein Transfer Learning eingesetzt, sondern die Gewichte zufällig initialisiert.

*inceptionResNet* steht für ein Inception-ResNet V2 [97], das mit einer Bildauflösung von  $448 \times 448$  trainiert wurde und bei *resnet50\_512* handelt es sich um ein ResNet-50 mit Bildauflösungen von  $512 \times 512$ .

Der Bezeichner *mobNet\_1\_224* beschreibt ein MobileNet V2 mit Bildauflösung  $224 \times 224$  und für *mobNet\_2\_320* wurden Bilder der Auflösung  $512 \times 512$  mittels Template Matching und Skalierung auf  $320 \times 320$  skaliert. Die gleiche Vorverarbeitungsstrategie wurde für die DenseNet-169 *dense\_169\_1\_320* und *dense\_169\_3\_320* sowie das MobileNet V2 *mobNet\_4\_322* angewendet. Die beiden letzten wurden ohne Transfer Learning trainiert.

Der Focal-Loss wurde für das DenseNet-121 *dense121\_6* und die MobileNet V2-Modelle *mobNet\_3* und *mobNet\_6* sowie das MobileNetV3 *mobNet\_5* genutzt. Für *mobNet\_6* wurde außerdem U-Ones+LSR verwendet. Das DenseNet-169 *dense169\_2* wurde mit zufälligen Gewichten initialisiert. *squeezezenet* (Squeezezenet-Architektur [120]), *shufflenet* (Shufflenet-Architektur [123]), *resNext\_1* (ResNext-50 ( $32 \times 4d$ )-Architektur [121]) und *wideResNet\_1*, *wideResNet\_2* (WideResNet-Architektur [124]) verwenden andere Modellarchitekturen, alle mit der Bildauflösung  $512 \times 512$ . *wideResNet\_2* wurde zusätzlich mit der U-Ones+LSR Strategie trainiert.

Modelle deren Bezeichner „dacm“ enthält, wurden mit dem DAM-Loss [102] trainiert. *dacm\_dense121* ist ein solches Modell der DenseNet-121 Architektur mit Bildgröße  $512 \times 512$ . Analog ist *dacm\_dense121\_320* ein DenseNet-121 und *dacm\_dense169\_320* ein DenseNet-169 mit Auflösung  $320 \times 320$ . Mit der Inputgröße  $224 \times 224$  wurden hingegen ein DenseNet-201 (*dacm\_dense201\_224*), ein Xception [98] (*dacm\_xception\_224*), ein ResNet-50 (*dacm\_resnet50\_224*) sowie ein ResNext-50 ( $32 \times 4d$ ) (*dacm\_resnext\_224*) trainiert.

Bei *logReg\_BR* und *logReg\_CC* handelt es sich um die Logistischen Regressionsmodelle, die die BR bzw. CC Problemtransformation sowie die PCA Dimensionsreduktion nutzen.

Die Methode des hierarchischen Lernens von Pham et al. [95] findet sich in den Modellen *hierar\_dense121*, *hierar\_xception* und *hierar\_nasnet* wieder, die die Architekturen DenseNet-121, Xception und NASNetAMobile (4 @ 1056) [99] verwenden.

*dacm\_dense121\_smooth* und *dacm\_dense121\_smoothV2* sind DenseNets-121, die beide mit Thorax-Röntgenbildern der Größe  $512 \times 512$  unter Anwendung des DAM-Loss und mit U-Ones+LSR bzw. LSR-V2 trainiert. Das MobileNet V2 *dacm\_mobNet\_prefocal* nutzt die gleiche Auflösung und ebenfalls den DAM-Loss, wurde allerdings mit dem Focal-Loss vortrainiert.

Bei *dacm\_xception\_320* und *dacm\_resNet50\_320* handelt es sich um ein Xception und

Pathologie	Bestes Einzelmodel	AUC
Atelectasis	DenseNet169, DAM-Loss, 512\x512, LSR-V2	0.897
Cardiomegaly	DenseNet169, DAM-Loss, 512\x512	0.888
Edema	DenseNet169, DAM-Loss, 512\x512	0.952
Consolidation	DenseNet169, DAM-Loss, 512\x512	0.950
Pleural Effusion	DenseNet121, DAM-Loss, 320\x320	0.948

**Tabelle 10.1:** Beste AUC-Werte eines einzelnen Modells der gesamten Modellbibliothek für die Vergleichslabel.

ResNet-50 Modell, die mit der Bildauflösung  $320 \times 320$  trainiert wurden. Dagegen bezeichnen *dacm\_dense169\_512* und *dacm\_resNet50\_512* ein DenseNet-169 respektive ResNet-50, die  $512 \times 512$  Aufnahmen verwenden und *dacm\_dense201\_480* ein DenseNet-201 mit  $480 \times 480$ . Letzteres konnte aufgrund der eingesetzten Grafikkarte nicht mit  $512 \times 512$  Bildern trainiert werden. *dacm\_dense169\_smoothV2\_512* ist ein DenseNet-169, das mit LSR-V2 trainiert wurde und *dacm\_dense169\_scratch\_512* wurde mit zufälligen Gewichten initialisiert.

Bei *dacm\_dense169\_horiz{0, 1, 2, 3}\_512* handelt es sich um Densenets-169 die zunächst durch Snapshotting entstanden sind und anschließend mit dem DAM-Loss weiter trainiert wurden.

*dacm\_mutli\_mobNet* und *dacm\_mutli\_dense121* wurden mit der aktualisierten LibAUC-Bibliothek [104] trainiert, die Multi-Label Klassifikation unterstützt.

In Tabelle 10.1 wird aufgezeigt, welches Modell der Bibliothek für die jeweilige Pathologie den höchsten AUC-Wert auf den Validierungsdaten erreicht. Dabei ist äußerst auffällig, dass diese Modelle alle mit dem DAM-Loss trainiert wurden.

Das Verfahren zur Ensemble-Selektion wurde mit unterschiedlichen Ensemble-Größen sowohl für das Ziehen mit als auch für das Ziehen ohne Zurücklegen durchgeführt. Dafür wurde mit jedem Modell die Vorhersage für den Validierungsdatensatz erzeugt und dann der Algorithmus aus [38] angewendet.

In Tabelle 10.2 sind die Ergebnisse für die verschiedenen Ensembles aufgeführt. Diese zeigt, dass ein Vergrößern des Ensembles nur zu geringen Verbesserungen führte. Ab etwa fünf Modellen bringt das weitere Hinzufügen keine relevanten AUC-Erhöhungen. Wird ohne Zurücklegen gezogen kann es passieren, dass durch den Algorithmus ab einem gewissen Iterationsschritt auch Modelle hinzugefügt werden, die die Gesamtperformance verschlechtern. Die Vorhersage des Ensembles wurde dabei als empirisches Mittel der jeweiligen Vorhersagen der Einzelmodelle, also den Wahrscheinlichkeiten, dass die Pathologie vorliegt, berechnet.

Anzahl Modelle	AUC, mit Zurücklegen	AUC, ohne Zurücklegen
2	0.9284	0.9284
3	0.9291	0.9281
4	0.9297	0.9286
5	0.9306	0.9295
10	0.9308	0.9294

**Tabelle 10.2:** Vergleich der AUC-Werte auf den Validierungsdaten mit einem Ensemble das  $n$  Modelle enthält und bei dem diese mit bzw. ohne Zurücklegen gezogen wurden.

## 10.3 Analyse der Korrelation

Da ausreichend Diversität ein wichtiges Kriterium für eine erfolgreiche Ensemble-Bildung ist, kann es sinnvoll sein, die Korrelation der Modelle zu untersuchen. Eine Darstellung der Korrelation aller Ensemble-Kandidaten befindet sich aufgeschlüsselt nach den Labeln als Matrizen in den Abbildungen B.17 bis B.21. Als Maß für die Korrelation wurde der Korrelationskoeffizient nach Bravias-Pearson verwendet. Der Korrelationskoeffizient  $r$  zwischen den beiden Modellvorhersagen  $x$  und  $y$  für  $n$  Bilder berechnet sich als

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \quad (10.1)$$

wobei  $\bar{x}$  und  $\bar{y}$  je die arithmetischen Mittelwerte von  $x$  und  $y$  sind [33, S. 118].

Der Wertebereich dieses Koeffizienten liegt bei  $-1 < r < 1$  und das Vorzeichen zeigt an, ob eine positive oder negative Korrelation vorliegt [33, S. 118]. Eine positive Korrelation würde bedeuten, dass mit steigendem  $x$  auch  $y$  steigt, respektive mit sinkendem  $x$  auch  $y$  sinkt. Dagegen weist ein negativer Korrelationskoeffizient auf entgegen gerichtetes Verhalten hin.

Die Korrelation zwischen den Vorhersagen zweier Modelle kann auch im Kontext der Überlegungen aus Abschnitt 2.5 gesehen werden. Eine sehr hohe positive Korrelation wäre gleichbedeutend mit ähnlichen Modellvorhersagen, und somit auch ähnlichen Fehlern. Das Ensembling hätte also einen geringeren positiven Effekt.

## 10.4 Finales Ensemble

Auf Basis der Ergebnisse, die in Tabelle 10.2 aufgeführt sind, wurde die Entscheidung getroffen, ein Ensemble, das fünf Modelle beinhaltet und das durch Ziehen mit Zurücklegen erzeugt wurde, als finales Ensemble zu wählen. Bei dieser Entscheidung wurde berücksichtigt, dass eine größere Modellanzahl auch größeren Aufwand für eine etwaige Bereitstellung und die Einreichung bei der Competition (vgl. Kapitel 11) nach sich zieht.

Durch den Algorithmus aus [38] wurden folgende Modelle selektiert: zwei Mal das DenseNet-169 und ein ResNet-50, die mit Bildauflösung  $512 \times 512$  und dem DAM-Vorgehen aus [102] trainiert wurden, ein weiteres DenseNet-169 mit  $512 \times 512$ , das mit der LSR-V2 Methode vorgenommen und anschließend mit dem DAM-Loss Fine-tuned wurde und das DenseNet-169, das aus dem Experiment zum Snapshotting stammt und mit DAM Fine-tuned wurde (vgl. Snapshot Nr. 1 aus Tabelle 9.11) und als Bildgröße  $320 \times 320$  verwendet.

In Abbildung 10.1 werden die Korrelationsmatrizen zwischen den Ensemble-Mitgliedern je Label visualisiert. Daran ist erkennbar, dass die Korrelation der Modelle je nach Pathologie unterschiedlich stark ist. Die Korrelationen sind allesamt positiv. Generell ist zu erwarten, dass die Modelle hinreichend stark korrelieren, da sie individuell alle ähnlich gute AUC-Werte erreichen. Insgesamt sind die Korrelationen bei den Pathologien Pleural Effusion und Edema am stärksten. Auch bei Cardiomegaly liegen starke Korrelationen vor, wobei jedoch das ResNet-50 heraussticht, da es deutlich weniger stark mit den anderen Modellen korreliert. Für das Label Atelectasis ist die Korrelation durchschnittlich am geringsten.

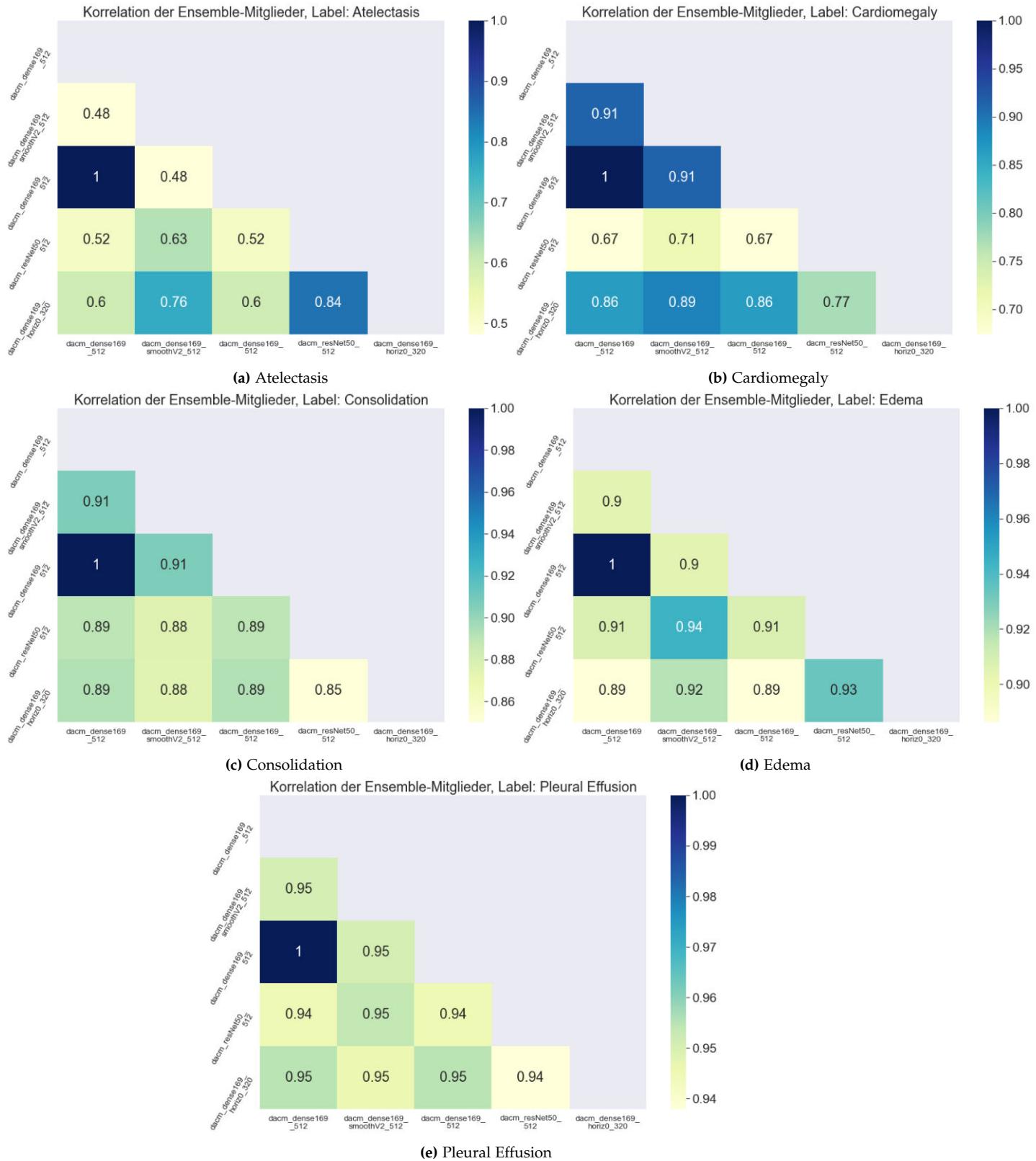
Nachdem die Modelle für das Ensembling ausgewählt wurden, können neben dem Mittelwertbilden auch andere Ensembling-Techniken zur Kombination der Modelle angewendet werden. Dafür wurden eine Logistische Regression, eine SVM mit linearem Kernel sowie ein AdaBoostClassifier verwendet. Der AdaBoostClassifier wurde mit je 25 Klassifizierern des Typs DecisionTreeClassifier der Tiefe eins und Logistischen Regression als Basisklassifizierer genutzt. Das Vorgehen, das ein Modell trainiert wird, das die Kombination der Ensemble-Mitglieder erlernen soll, entspricht dem Stacking (vgl. Abschnitt 2.5.2).

Dabei wurde je ein Modell des entsprechenden Types für jede Pathologie verwendet, das die beste Kombination der CNNs erlernen soll. Für das Fitten dieser Klassifizierer wurden nur Trainingsdaten verwendet, bei denen die fünf Vergleichspathologien keine uncertain Label enthalten. Dies trifft auf etwa 160.000 Bilder zu, von denen 150.000 gesampled wurden. Mit allen Ensemble-Mitgliedern wurde ein Forward-Pass für diese Daten ausgeführt, sodass deren Vorhersagen als Trainingsdaten für den Stacking-Klassifizierer dienen können. Die Ergebnisse sind in Tabelle 10.3 aufgeführt.

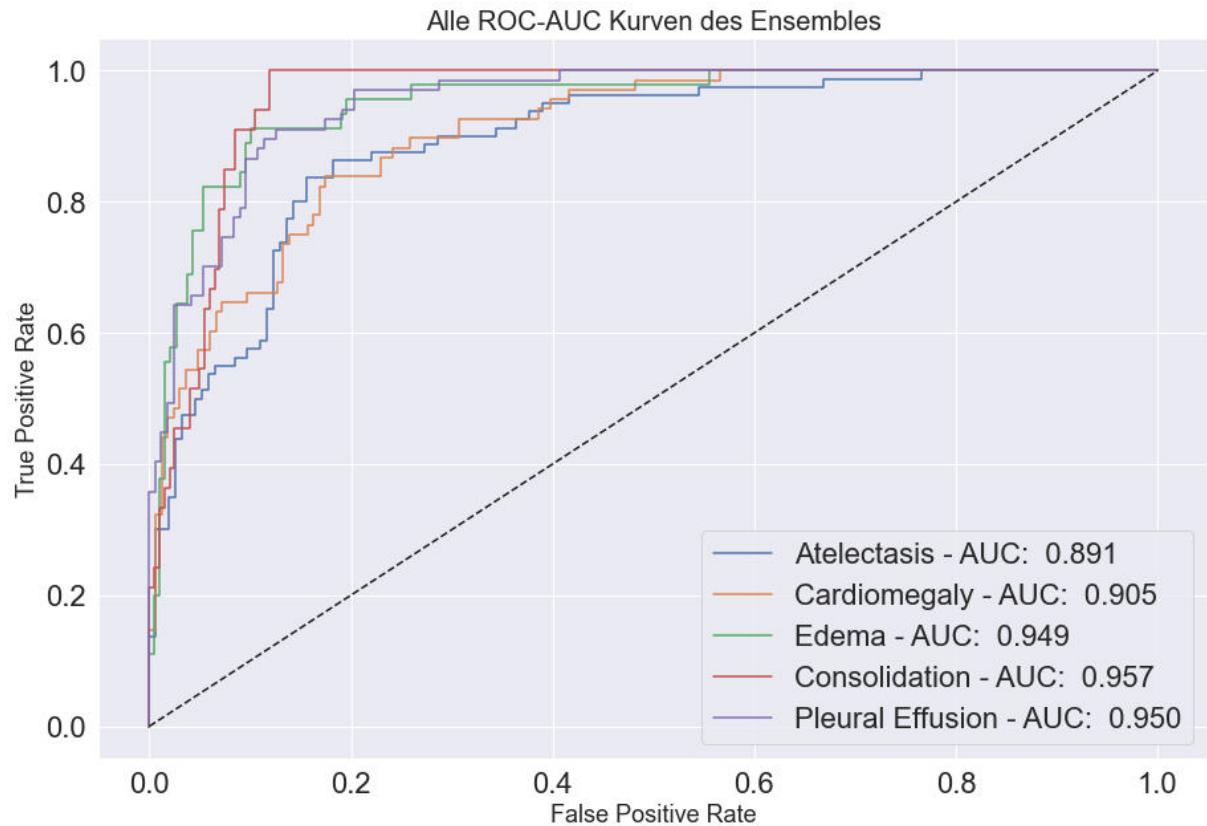
Dabei ist zu erkennen, dass die SVM sehr schlechte Ergebnisse liefert und das AdaBoost-Verfahren mit einer Logistischen Regression die besten AUC-Werte auf den Validierungsdaten erzeugt. Selbst diese liegen jedoch etwas unter der einfachen Mittelwertbildung. Dabei ist allerdings berücksichtigen, dass die Ensemble-Kandidaten auch anhand dieses Kriteriums ausgewählt wurden, was den Vergleich verzerrt.

Da die Verwendung eines zusätzlichen Stacking-Klassifizierers keine Verbesserung brachte, wurde die Entscheidung getroffen, nur ein Soft Voting durchzuführen. Auch mehrere einfache Neuronale Netze wurden getestet. Diese konnten jedoch ebenfalls zu keiner Verbesserung führen.

Die ROC-Kurven inklusive den AUC-Werten des Ensembles sind in Abbildung 10.2 dargestellt. Außerdem lässt sich der AUC-Wert der einzelnen Pathologien mit denen von Pham et al. [95] vergleichen (siehe Tabelle 10.4), da diese in der Veröffentlichung angegeben werden. Der Gesamtunterschied beträgt rund 1%.



**Abbildung 10.1:** Korrelation der fünf Ensemblemitglieder, die mit Zurücklegen gezogen wurden.



**Abbildung 10.2:** ROC-AUC-Kurven des gewählten Ensembles. Während die Pathologien Edema, Consolidation und Pleural Effusion alle sehr gute AUC-Werte von etwa 0.95 erreichen, sind die Ergebnisse für Atelectasis und Cardiomegaly mit einem AUC von etwa 0.9 rund 5% schlechter.

Stacking Modell	AUC Trainingsdaten	AUC Validierungsdaten
Logistische Regression	0.834	0.9084
SVM	0.620	0.587
AdaBoost(DT)	0.834	0.908
AdaBoost(LR)	0.830	0.928

**Tabelle 10.3:** AUC Werte für die Stacking-Klassifizierer auf den Trainings- und Validierungsdaten. Abkürzungen: *DT*: DecisionTree, *LR*: Logistische Regression.

Pathologie	AUC Ensemble	AUC aus [95]	Differenz
Atelectasis	0.891	0.909	0.018
Cardiomegaly	0.905	0.910	0.005
Edema	0.949	0.958	0.009
Consolidation	0.957	0.957	0
Pleural Effusion	0.950	0.964	0.014
Mittelwert	0.930	0.940	0.01

**Tabelle 10.4:** AUC Werte des Ensembles für die einzelnen Pathologien im Vergleich zu Pham et al. [95]. Die größte Differenz besteht für die Label Atelectasis und Pleural Effusion. Im Mittel über alle Pathologien entspricht der Unterschied etwa 1%.

## Kapitel 11

# Einreichung bei der CheXpert-Competition

Die Autoren des CheXpert-Datensatzes haben mit dessen Veröffentlichung auch einen Wettkampf gestartet, die CheXpert-Competition [15]. Die Einreichung geschieht dabei über ein Codalab Worksheet [125] und die Datensatzautoren stellen eine Anleitung [126] dafür bereit. Dabei sind allerdings einige Hindernisse zu berücksichtigen, die zum Teil nicht direkt aus der Anleitung klar werden. Daher wird das für diese Arbeit verwendete Vorgehen im Folgenden beschrieben.

Zunächst kann den Schritten der Anleitung gefolgt werden. Es wird demnach ein neues Codalab Worksheet erstellt und die folgenden Befehle ausgeführt:

```
1 cl add bundle chexpert-utils//valid .
2 cl add bundle chexpert-utils//valid_image_paths.csv .
```

Um das eingereichte Ensemble überprüfen zu können, wurde ein Python-Skript erzeugt, das als Kommandozeilenargumente den Pfad zu einer CSV-Datei für den Input und einer für den Output verarbeitet. Dieses Skript definiert zuerst die nötigen Modelle und lädt deren trainierten Gewichte. Die Gewichte wurden im gleichen Verzeichnis abgelegt wie das Skript und die daraus erstellte zip-Datei zuvor hochgeladen. Beim Laden der Gewichte ist zu berücksichtigen, dass den Dateipfaden ein `src/` vorangestellt werden muss.

Es wurde eine einfache `dataset`-Klasse implementiert, mit der dann der `Dataloader` initialisiert wurde. Für diesen wurde dann ein Forward-Pass ausgeführt um die Modell-Vorhersagen zu erzeugen. Im folgenden Schritt besteht allerdings eine Besonderheit bei der Modellevaluation. Die Richtlinien geben vor, dass nicht für jedes Bild eine Vorhersage abgegeben werden soll, sondern für jede Studie. Eine Studie kann dabei mehrere frontale und / oder laterale Aufnahmen enthalten.

In der vorliegenden Arbeit wurde der Mittelwert aller Vorhersagen innerhalb einer Studie als Studienvorhersage verwendet. An dieser Stelle besteht allerdings weiteres Potenzial zur Optimierung des erzielten AUC-Wertes. So erzielte das Ensemble auf den frontalen Validierungsdaten einen AUC von 0.929 und auf den lateralen Validierungsdaten einen AUC von 0.907. Die Bilder des Validierungsdatensatzes stammen

alle aus unterschiedlichen Studien. Beim erzeugen der CSV, die die finalen Resultate enthält, muss berücksichtigt werden, dass anders als in der Anleitung beschrieben, zwischen den einzelnen Spalten keine Leerzeichen vorhanden sein dürfen, da dies zu Fehlern führt. Das Skript endet damit, dass die Datei unter dem als Kommandozeilenargument angegebenen Pfad gespeichert wird. Außerdem wurde bei dem verwendeten Docker-Image von der Anleitung abgewichen. Diese empfiehlt die Verwendung von `anibali/pytorch:cuda-9.0`. Dieses Image ist jedoch veraltet und es fehlen essenzielle Bibliotheken wie `pandas`, die auch nicht nachgeladen werden können. Stattdessen wurde `codalab/default-gpu` verwendet. Mit dem Skriptnamen `submitCheX.py` lautet der entscheidende Befehl:

```
1 cl run valid_image_paths.csv:valid_image_paths.csv CheXpert-v1.0:valid
src:SubmitCheX "python src/submitCheX.py valid_image_paths.csv
predictions.csv" -n run-predictions --request-docker-image codalab/
default-gpu --request-gpus 1 --request-memory 8g
```

Anschließend kann weiter nach der Anleitung verfahren werden.

## Kapitel 12

# Evaluierung auf externen Daten

Um die Generalisierungsfähigkeit der Modelle auf externen Daten beurteilen zu können, wurde das Ensemble auf dem ChestX-ray14-Datensatz evaluiert. Die Resultate auf diesen Testdaten befinden sich in Tabelle 12.1. Diese zeigen, dass die Modelle auch ohne Training auf den Daten durchaus Vorhersagen erzeugen, die zumindest mit der Baseline von Wang et al. [60] annähernd mithalten können. Jedoch performt das spezialisiertere Vorgehen von Guan et al. [100] deutlich besser.

Um ein mögliches Praxisszenario zu imitieren wurden weitere Experimente auf diesem Datensatz durchgeführt. Für die folgenden Schritte wurde das Einzelmodell mit dem schlechtesten AUC-Wert, also das DenseNet-169 mit Bildauflösung  $320 \times 320$ , verwendet.

Es wurde festgelegt, dass 5.000 zufällig aus den Trainingsdaten gezogene Bilder für das Training verwendet werden können. Damit soll eine Situation repräsentiert werden, in der ein Krankenhaus eine begrenzte Menge an Bildern labelt, um ein Modell auf die lokalen Gegebenheiten zu Fine-tunen. 20% dieser Trainingsdaten wurden als Validierungsdaten verwendet und bei jedem Trainingsdurchlauf jenes Modell

Modell / Methode	AUC
DenseNet169	0.741
DenseNet169, LSR-V2	0.746
ResNet50	0.721
Dense169, $320 \times 320$	0.714
Ensemble	0.754
Baseline [60]	0.770
AG-CNN [100]	0.893

**Tabelle 12.1:** AUC-Werte des finalen Ensembles sowie der Baseline von Wang et al. [60] und der AG-CNN Methode von Guan et al. [100] auf den Testdaten des ChestX-ray14-Datensatzes. Für die Vergleiche mit [60] und [100] wurde der mittleren AUC der Pathologien berechnet, wie sie in [100] angegeben werden.

Pathologie	Training CheXpert	Pretrain CheXpert, Fine-tune	Pretrain ImageNet, Fine-tune
Atelectasis	0.670	0.693	0.638
Cardiomegaly	0.756	0.773	0.628
Edema	0.737	0.742	0.703
Consolidation	0.649	0.660	0.594
Pleural Effusion	0.759	0.762	0.698
Durchschnitt	0.714	0.726	0.652

**Tabelle 12.2:** Vergleich der Evaluierung auf dem ChestX-ray14-Datensatz ohne Fine-tuning, mit Fine-tuning, sowie mit Fine-tuning ausgehend von ImageNet Gewichten. Zum Fine-tunen stand wurde ein Datensatz von 5.000 Bildern zufällig aus den Trainingsdaten gezogen.

ausgewählt, das auf den Validierungsdaten den höchsten AUC Wert erreichte. Es wurde für fünf Epochen trainiert. Als Verlustfunktion kam der DAM-Loss zum Einsatz, mit einer initialen Lernrate von 0.1, die nach der ersten und dritten Epoche um den Faktor zehn reduziert wurde.

Mit dieser Trainingskonfiguration wurden das auf dem CheXpert-Datensatz vortrainierte Modell, sowie eines mit auf dem ImageNet Datensatz vortrainierten Gewichten, trainiert. Für letzteres wurde auch getestet eine deutlich höhere Epochenzahl zu trainieren (100 Epochen), diese führte jedoch zu keinen Verbesserungen. Die Ergebnisse werden in Tabelle 12.2 aufgeführt. Daraus wird ersichtlich, dass mit einer vergleichsweise geringen Datenmenge das Finetuning auch nur zu einer leicht höheren AUC geführt hat. Im Vergleich zu der Initialisierung mit den ImageNet Gewichten stellt dies jedoch eine Verbesserung von etwa 10% dar.

# Kapitel 13

## Resultate

In der vorliegenden Arbeit wurden mehrere Methoden aus der Literatur analysiert und mit weiteren Ansätzen zur Verbesserung der Modellvorhersagen erprobt. Dabei wurden Maßnahmen zu den drei großen Bereichen Datensatz (u.a. Imbalance, uncertain Label, Label-Hierarchie), Modell (u.a. Architektur, Gewichtsinitialisierung, Feature Extraction) und Training (u.a. Conditional Training, Verlustfunktionen, Intra-Epoch Evaluation) untersucht.

Die Resultate waren gemischt. Während einige Methoden sehr gut funktionierten, z.B. FRODO [91] und der DAM-Loss von Yuan et al. [102], bewährten sich manche nur teilweise (z.B. Label-Hierarchie basiertes Training von Pham et al. [95]) oder gar nicht, wie die Boosted Cascade von Kumar et al. [12]. Die AG-CNN Methode von Guan et al. [100] funktionierte prinzipiell zwar, jedoch führte sie aufgrund der Art, in der sich die fünf Vergleichspathologien auf Röntgenbildern äußern, zu keinen nennenswerten Verbesserungen.

Am erfolgreichsten war der DAM-Loss und die Intra-Epoch Evaluation, was sich auch am finalen Ensemble gut erkennen lässt. Zudem scheint die Erhöhung der Bildauflösung ebenfalls Vorteile zu bringen. Jedoch ist hierbei zu berücksichtigen, dass für andere Pathologien dies nicht zwingend zutreffen muss (vgl. Tang et al. [71]).

Das Ausgleichen der Imbalance der Trainingsdaten erwies sich nicht als zielführend. Stattdessen scheint die Gewichtung der Verlustfunktion sinnvoller zu sein.

Die Behandlung der uncertain Label beeinflusst die Modellgüte stark. Mehrere Ansätze wie die U-Ones+LSR, LSR-V2 und U-Mixed sind vielversprechend und eine Kombination in Form eines Ensembles ist naheliegend.

Das Template Matching funktionierte dahingehend gut, dass die auszuschneidenden Lungenbereiche erkannt wurden, jedoch konnten keine signifikanten Vorteile für das Modelltraining oder Ensembling festgestellt werden.

Während die Bilddatenverarbeitung allgemein wichtig war für gute Trainingsergebnisse, führte die Kontrastverbesserung mittels CLAHE zu keinen weiteren Verbesserungen der Metrik.

Das Verwenden unterschiedlicher Verlustfunktionen beeinflusst das Training, wie zu erwarten ist, in großem Maß. Der Focal-Loss, der ursprünglich nicht für die Bildklassifikation gedacht war, bietet sich wegen der Klassen-Imbalance an und führte zu

vergleichbaren Ergebnissen wie der BCE Loss. Dahingegen erwies sich der LSEP bzw. Smooth PWE-Loss als ungeeignet.

Wird ein CNN als Feature Extractor verwendet, kann der Einsatz einer PCA zu einer erfolgreichen Dimensionsreduktion führen. Bei anschließender Problemtransformation erwiesen sich die BR- und die CC-Transformation als in gleicher Weise geeignet.

Das Snapshotting brachte gute Ergebnisse und kann besonders mit der Gewichtsreinitialisierung genutzt werden, um mit geringeren Ressourcen Ensembling-Effekte zu nutzen. Vor dem Trainingsbeginn kann statt einer Initialisierung mit den ImageNet Gewichten auch eine weight transfusion in Betracht gezogen werden. Bezuglich der CNN-Architekturen lässt sich feststellen, dass die DenseNet-169-Architektur am erfolgreichsten war, was sich sowohl an Tabelle 10.1 als auch an deren mehrfachen Vorkommen im finalen Ensemble erkennen lässt. Nichtsdestotrotz ist beachtenswert, dass die MobileNet V2 Architektur, die deutlich kleiner und daher schneller ist, ähnlich gute Ergebnisse erzielt wie die DenseNet- oder ResNet-Architekturen.

Die Resultate zeigen insgesamt, dass die Besonderheiten der vorliegenden Problemstellung berücksichtigt werden müssen, und daher spezielle Maßnahmen notwendig sind, um erfolgreiche Modelle zu trainieren.

Das finale Ensemble wurde bei der CheXpert Competition eingereicht und erzielte auf dem geheimen Testset einen mittleren AUC von 0.917 auf den fünf Vergleichslabellen. Damit schlägt es durchschnittlich 2.2 der drei Vergleichsradiologen und belegt Platz 54. Die ersten zehn Plätze der Rangliste, wie sie auf [15] eingesehen werden können, sind auch in Tabelle 13.1 aufgeführt. Das in dieser Arbeit erstellte Ensemble ist auf [15] unter der Modellbezeichnung „sebstein“ aufgeführt. Die ROC-AUC des finalen Ensembles ist somit um rund 1.4% schlechter als die des Erstplatzierten.

Auffällig an der Rangliste ist, dass die Gruppe bzw. Methode von Pham et al.[95] sechs der ersten 10 Plätze belegt. Die Plätze neun und zehn sind ebenfalls vom selben Autor, allerdings gibt es dazu keine Veröffentlichung.

Rang	Modellbezeichnung (Referenz)	AUC	Anzahl Radiolo- gen unter ROC- Kurve
1	DeepAUC-v1 ([102], Kapitel 7.7)	0.930	2.8
2	Hierarchical- Learning-V1 ([13], Kapitel 7.3)	0.930	2.6
3	Conditional- Training-LSR ([13], Kapitel 7.3)	0.929	2.6
4	Hierarchical- Learning-V4 ([13], Kapitel 7.3)	0.929	2.6
5	YWW ([76], Kapitel 6.1.3)	0.929	2.8
6	Conditional- Training-LSR-V1 ([13], Kapitel 7.3)	0.929	2.6
7	Hierarchical- Learning-V0 ([13], Kapitel 7.3)	0.929	2.6
8	Multi-Stage- Learning-CNN-V3 ([13], Kapitel 7.3)	0.928	2.6
9	DeepCNNsGM	0.928	2.6
10	DeepCNNs	0.927	2.6
54	sebstein	0.917	2.2

**Tabelle 13.1:** Auszug der Rangliste der CheXpert Competition. Bei den Methoden, die auch in dieser Arbeit verwendet oder vorgestellt wurden, sind die Referenzen angegeben.

# Kapitel 14

## Limitationen

Wenngleich die in dieser Arbeit vorgestellten Methoden zum Teil sehr gute Ergebnisse erzielen, existieren dennoch einige Hürden und Limitationen, die im Folgenden aufgezeigt werden. Diese werden in technische sowie praxisbezogene Limitationen unterteilt. Letztere lassen sich zum Teil auch allgemein auf den Einsatz von KI-basierten CAD-Systemen beziehen.

### 14.1 Technische Limitationen

Als eine relevante technische Limitation können die vorhandenen Ressourcen beziehungsweise die verfügbare Hardware gesehen werden. Um die hohe Auflösung, die medizinische Bilder aufweisen, vollständig nutzen zu können, wäre eine sehr große Menge an unter anderem (GPU-)Speicher notwendig.

Das Ensembling mehrerer Ansätze führt zudem zu höherem Implementierungsaufwand, besonders wenn nicht auf Bibliotheken zurückgegriffen werden kann. Dies bringt zudem Herausforderungen für das Arbeiten in etwaigen Entwicklungsteams mit, da die gemeinsame Nutzung von Jupyter Notebooks in der Praxis häufig zu Schwierigkeiten beispielsweise bezüglich der Versionsverwaltung führt. Ein ähnlich gelagertes Problem können wachsende Datensätze darstellen, wenn beispielsweise neue Krankheiten oder Bilder hinzukommen. Während es für die Versionsverwaltung von Programmiercode vielfältige Lösungen gibt, ist die Versionierung von Modellen und den zugehörigen Datensätzen noch nicht derartig gut erschlossen.

Bezüglich der Auslieferung derartiger Systeme kann die Containerisierung große Vorteile bieten, die auch ein institutionsübergreifendes Deployment möglich macht. Bereits die CheXpert Competition zeigt, dass durch geeignete (Docker) Container ein reproduzierbares Verhalten sichergestellt werden kann. Jedoch ist für einen erfolgreichen Praxiseinsatz auch eine passende Visualisierung nötig. Somit ist für die Umsetzung von derartigen KI-Projekten Fachwissen aus verschiedenen Bereichen (KI, IT-Sicherheit, Schnittstellenanbindung, Containerisierung/Deployment, Visualisierung, etc.) notwendig, was den Einsatz eines Entwicklungsteams unabdingbar macht.

## 14.2 Praxisbezogene Limitationen

Während der Einsatz von KI zur Krankheitsklassifikation ein sehr fruchtbarer Forschungsbereich ist, existieren für den tatsächlichen Praxiseinsatz von CAD-Systemen zur Analyse von Thorax-Röntgenbildern vielfältige Hürden, die im Folgenden zusammengefasst werden.

### 14.2.1 Datensatz

Einige dieser Limitationen ergeben sich aus den (öffentliche) verfügbaren Datensätzen und ihrer Entstehungsweise.

*Datensatz-Größe:* Wenngleich in den letzten Jahren die Anzahl an öffentlich verfügbaren Bildern deutlich gestiegen ist, sollte bedacht werden, dass für zugelassene Systeme (vgl. Abschnitt 2.2) die verwendete Datenmenge etwa 1.000 mal so groß ist, wie der größte öffentliche Datensatz (MIMIC-CXR). Außerdem zeigt die Beschreibung von Annarumma et al. [11] zur Erstellung ihres Datensatzes, bei dem von 830.000 gesammelten Bildern nur 470.000 für das Modelltraining verwendet werden konnten, dass nicht zwingend alle verfügbaren Aufnahmen geeignet sind.

*Labeling:* Das NLP- / regelbasierte automatische Erzeugen der Label birgt gewisse Gefahren bezüglich der Gütebeurteilung. So wurde in dieser Arbeit festgestellt, dass der CheXpert-Datensatz rund 4.000 Bilder enthält, bei denen zwingend ein Fehler im Labeling vorliegen muss (vgl. Kapitel 8.8.2). Calli et al. [10] betonen, dass die Gütebewertung anhand von durch Radiologen gelabelten Goldstandard-Daten erfolgen sollte. Dies führt jedoch zu einem Dilemma zwischen der Anzahl an Testdaten und dem Labeling-Aufwand. Johnson et al. [65], die Autoren des MIMIC-CXR-Datensatzes, führen auch einen Vergleich zwischen dem CheXpert-Labeling-Tool und einem weiteren Tool, NegBio [63], an. Der Mittelwert über alle Pathologien des prozentualen Anteils an Bildern, für denen die extrahierten Label der beiden Tools nicht übereinstimmen, beträgt 0.8% mit einer Standardabweichung von 0.65%. Der größte Anteil an Differenzen besteht mit 2.6% für Cardiomegaly und die geringste mit 0% für Pleural Other. Dies deutet darauf hin, dass das CheXpert-Labeling-Tool insgesamt sehr nah am aktuellen Entwicklungsstand ist.

*Bias:* Darüber hinaus sollte der Bias, also Verzerrungen, innerhalb der Datensätze berücksichtigt werden. Dieser kann sich mannigfaltig äußern. Machine Learning Modelle erlernen und propagieren ohne gesonderte Maßnahmen im Allgemeinen einen den Trainingsdaten inhärenten Bias [127]. Seyyed et al. [128] untersuchten die Datensätze CheXpert, MIMIC-CXR und ChestX-ray14 bezüglich ihres Bias. Sie kamen zu dem Ergebnis, dass diese Datensätze für Männer ab 40 Jahren vorteilhaft sind. Für die Kombination der Datensätze war der Bias hinsichtlich Alter und Geschlecht am geringsten. Die Fusion mehrere Datenquellen könnte demnach ein entscheidender Baustein in Richtung „fairer“ CAD-Systeme sein. Der MIMIC-CXR-Datensatz liefert zudem Informationen zur ethnischen Herkunft und dem Versicherungstyp der Patienten. Daraus ergab sich, dass die Unterrepräsentierung von sozioökonomisch Schwachen sich negativ auf deren Krankheitserkennung (TPR) auswirkt [128]. So wurden in den Expe-

rimenten Hispanoamerikaner und Medicaid-Versicherte benachteiligt [128]. Medicaid ist ein Versicherungsprogramm für US-Amerikaner mit geringem Einkommen [129].

*Datensatz-Qualität:* Ein Aspekt der generell in Erwägung gezogen werden muss, ist die Qualität der genutzten Daten. Die Bildqualität ist wie in Abschnitt 8.1 dargestellt zumindest für den CheXpert-Datensatz stellenweise sehr gering. Für den in der vorliegenden Arbeit bearbeiteten Anwendungsfall stellte sich dies allerdings nicht als besonders problematisch heraus.

### 14.2.2 Einsatz in Krankenhäusern

Auch neben der Güte der Modelle und derer Trainingsbedingungen steht der Einsatz in Krankenhäusern vor weiteren Hürden.

*Vorkommende Krankheiten:* Ein Problem könnte entstehen, wenn ein eingesetztes CAD-Systeme kein ausreichend großes Spektrum an für den Praxisalltag relevanten Krankheiten abdeckt. Dieses muss sich auch in den Trainingsdaten widerspiegeln. Ansonsten muss davon ausgegangen werden, dass Krankheiten, die nicht als Trainingslabel vorkamen, falsch klassifiziert werden. Beispielsweise könnte eine COVID-19 Erkrankung mit keinem der großen öffentlichen Datensätzen festgestellt werden. Dies wäre erwartungsgemäß auch schwierig mit einer automatisierten OoD abzufangen, da die Ähnlichkeit der Röntgenbilder auch bei unbekannten Krankheiten durchaus hoch genug sein kann.

*Bestehende Arbeitsweisen:* Der Arbeitsalltag in Krankenhäusern ist bestimmt durch politische, wirtschaftliche und gewohnheitsbedingte Faktoren [130]. KI-Systeme müssen darin integriert werden und könnten so laut [130] durch ihren bloßen Einsatz zu keinen großen Veränderungen führen.

*Lokale Besonderheiten:* Für die einzelnen Krankenhäuser können lokale Besonderheiten und geographische Variationen gelten, die nicht in den Trainingsdaten allgemeiner Modelle abgebildet sind. So kann es unter anderem zu einer Änderung der Patientenkohorten kommen, wenn es sich beispielsweise um ein Kinderkrankenhaus handelt. Auch die verwendeten Röntgengeräte können eine Verzerrung herbeiführen. Falls Variationen zwischen den einzelnen Stationen eines Krankenhauses vorliegen, kann es notwendig sein, diese zu berücksichtigen (vgl. Annarumma et al. [11]), wozu sehr spezifisches Wissen notwendig ist. Die Ergebnisse von Tang et al. [71] können so interpretiert werden, dass zumindest die Erkennung von abnormalen Röntgenbildern ausreichend robust gegenüber geographischen Variationen ist.

*Fehlende Dateninfrastruktur:* Zwar besteht technisch die Möglichkeit, Neuronale Netze mittels Fine-tuning an die lokalen Gegebenheiten eines Krankenhauses anzupassen und so negative Effekte des Bias durch beispielsweise Kohortenänderungen abzuschwächen. Dafür ist allerdings in den jeweiligen Krankenhäusern eine entsprechende Dateninfrastruktur notwendig, die zumeist nicht vorhanden ist [130]. Die Enquete-Komission „Künstliche Intelligenz – Gesellschaftliche Verantwortung und wirtschaftliche, soziale und ökologische Potenziale“ des Deutschen Bundestages [131, S. 243] weist darauf hin, dass auch in Deutschland die „Digitalisierung der Infrastruktur im Gesundheitsbereich“ beschleunigt werden sollte. Einheitliche Taxonomien, wie das Unified Medical

Language System (UMLS), das der PadChest-Datensatz [64] nutzt, könnten diese Prozesse unterstützen.

### 14.2.3 Rechtliche Limitationen

Für den Praxiseinsatz spielen zudem rechtliche Anforderungen eine Rolle, die zwar thematisch auch zu dem vorherigen Abschnitt passen, jedoch herausgehoben werden sollten.

*Datennutzung:* Die öffentlichen Datensätzen besitzen üblicherweise einen Sperrvermerk, der die kommerzielle Nutzung untersagt (vgl. beispielsweise das Research Use Agreement des CheXpert-Datensatzes [15]). Die Enquete-Kommision Künstliche Intelligenz empfiehlt, die Verfügbarkeit von medizinischen Daten zu Forschungszwecken zu verbessern, indem ein Register, eine freiwillige Datenfreigabe und offene Standards eingeführt werden, sowie die Gesetzgebung in Einklang mit der Datenschutz-Grundverordnung zu überarbeiten [131, S. 243].

*Zulassungsverfahren:* Für den Einsatz von CAD-Systemen sind entsprechende Zulassungen und Zertifikate, wie das € Zeichen, gesetzlich vorgeschrieben. Die derzeitigen Zulassungsverfahren sind jedoch nicht für die neuen Technologien geeignet und sollten daher angepasst werden [131, S. 243].

*Haftung:* Bezuglich der Haftung gibt es noch viel Unsicherheit und Handlungsbedarf. Dies wird klar durch die Forderung der Enquete-Komission an die Bundesregierung „[m]ögliche Lücken und Unsicherheiten bezüglich der Haftung bei der Anwendung von KI im Gesundheitswesen [...] zu ermitteln; wo notwendig [...] transparente Regelungen zu schaffen und Lücken bzw. Unsicherheiten mithilfe von Normierungs- und Standardisierungsverfahren zu beseitigen bzw. zu verringern“ [131, S. 244].

## Kapitel 15

# Zusammenfassung und Ausblick

In dieser Arbeit wurden diverse Methoden zur Krankheitsklassifikation auf Thorax-Röntgenbildern sowie deren Ensembling untersucht. Diese setzten zur Optimierung an unterschiedlichen Aspekten des Trainings an. Wie in Kapitel 13 dargelegt, kann keine pauschale Aussage über die Effektivität der ausgewählten Methoden getroffen werden.

Die in dieser Arbeit eingesetzten Methoden könnten in Anlehnung an Baltruschat et al. [16] um eine Bone suppression erweitert werden. Die Lung field segmentation ist hingegen sehr ähnlich zu dem getesteten TM, sodass die zusätzliche Implementierung nicht vielversprechend wirkt. Zudem muss hierbei berücksichtigt werden, dass im verwendeten Datensatz auch laterale Aufnahmen enthalten sind.

Bezüglich der Integration derartiger Tools in den Praxisalltag zeigt Lunit [23], dass eine Kooperation aus Unternehmen, die auf KI spezialisiert sind, mit Herstellern von Röntgengeräten vielversprechend sein kann. Zu den wichtigen Funktionen solcher Systeme zählt neben der reinen Klassifikation auch die Markierung von Läsionsbereichen (vgl. u.a. [76, 100]) als auch die Erzeugung textueller Berichte (vgl. [19, 23]).

Die in dieser Arbeit aufgeführten Limitationen zeigen allerdings, dass vollständig automatisierte Diagnosen aktuell zu optimistisch und unsicher sind. Stattdessen sollte der Fokus auf der Entwicklung von Human-in-the-loop-Systemen liegen, beziehungsweise die Systeme nur assistierend eingesetzt werden.

Angesichts des Labeling kann darüber diskutiert werden, ob eine Klassifikation nach normal – abnormal beziehungsweise gesund – ungesund, beispielsweise in Verbindung mit einer automatisierten Patienten-Priorisierung wie in Kapitel 7.2 beschrieben, für gewisse Praxiseinsätze erstrebenswerter ist als die Erkennung spezifischer Pathologien.

Aus Sicht der Forschung wird die COVID-19-Pandemie sicherlich weiter für viel Interesse und Auftrieb sorgen, besonders wenn der erste größere Datensatz dazu veröffentlicht wird. In diesem Forschungsgebiet sind in den nächsten Jahren sicherlich große Fortschritte zu erwarten. Ein Großteil der Publikationen basiert derzeit auf allgemeinen Methoden der Bildverarbeitung und eine Spezialisierung auf die Domäne der Thorax-Röntgenbilder ist nur vereinzelt festzustellen. Gleichzeitig ist unbestreitbar, dass im medizinischen Bereich sehr große Einsatzpotenziale für KI bestehen.

Besonders vor diesem Hintergrund werden für Deutschland die Forderungen der En-

quete-Kommission Künstliche Intelligenz [131] zentral, die Wege für die Digitalisierung der Gesundheitsbranche zu ebnen.

# Literatur

- [1] *Chest X-rays - Mayo Clinic*. URL: <https://www.mayoclinic.org/tests-procedures/chest-x-rays/about/pac-20393494> (besucht am 08.10.2021).
- [2] Cristina Mitchell und <https://www.facebook.com/pahowho>. *PAHO/WHO | World Radiography Day: Two-Thirds of the World's Population has no Access to Diagnostic Imaging*. Pan American Health Organization / World Health Organization. 8. Nov. 2012. URL: [https://www3.paho.org/hq/index.php?option=com\\_content&view=article&id=7410:2012-dia-radiografia-dos-tercios-poblacion-mundial-no-tiene-acceso-diagnostico-imagen&Itemid=1926&lang=en](https://www3.paho.org/hq/index.php?option=com_content&view=article&id=7410:2012-dia-radiografia-dos-tercios-poblacion-mundial-no-tiene-acceso-diagnostico-imagen&Itemid=1926&lang=en) (besucht am 08.10.2021).
- [3] Daniel J. Mollura u. a. „White Paper Report of the RAD-AID Conference on International Radiology for Developing Countries: Identifying Challenges, Opportunities, and Strategies for Imaging Services in the Developing World“. In: *Journal of the American College of Radiology* 7.7 (1. Juli 2010). Publisher: Elsevier, S. 495–500. doi: [10.1016/j.jacr.2010.01.018](https://doi.org/10.1016/j.jacr.2010.01.018).
- [4] *To X-ray or not to X-ray?* URL: <https://www.who.int/news-room/feature-stories/detail/to-x-ray-or-not-to-x-ray-> (besucht am 08.10.2021).
- [5] Pranav Rajpurkar u. a. „CheXnet: Radiologist-level pneumonia detection on chest x-rays with deep learning“. In: *arXiv preprint arXiv:1711.05225* (2017).
- [6] *Statistisches Bundesamt Deutschland - GENESIS-Online*. 24. Mai 2021. URL: <https://www-genesis.destatis.de/genesis/online> (besucht am 08.10.2021).
- [7] Forum of International Respiratory Societies und European Respiratory Society. *The global impact of respiratory disease*. OCLC: 999612837. 2017.
- [8] Utku Kose u. a. *Deep Learning for Medical Decision Support Systems*. Studies in Computational Intelligence. Springer Singapore, 2021. doi: [10.1007/978-981-15-6325-6](https://doi.org/10.1007/978-981-15-6325-6).
- [9] Bhavik N. Patel u. a. „Human-machine partnership with artificial intelligence for chest radiograph diagnosis“. In: *npj Digital Medicine* 2.1 (18. Nov. 2019). Number: 1 Publisher: Nature Publishing Group, S. 1–10. doi: [10.1038/s41746-019-0189-7](https://doi.org/10.1038/s41746-019-0189-7).
- [10] Erdi Çallı u. a. „Deep Learning for Chest X-ray Analysis: A Survey“. In: *Medical Image Analysis* (5. Juni 2021), S. 102125. doi: [10.1016/j.media.2021.102125](https://doi.org/10.1016/j.media.2021.102125).

- [11] Mauro Annarumma u. a. „Automated Triaging of Adult Chest Radiographs with Deep Artificial Neural Networks“. In: *Radiology* 291.1 (22. Jan. 2019). Publisher: Radiological Society of North America, S. 196–202. doi: [10.1148/radiol.2018180921](https://doi.org/10.1148/radiol.2018180921).
- [12] Pulkit Kumar, Monika Grewal und Muktabh Mayank Srivastava. „Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs“. In: *International Conference Image Analysis and Recognition*. Springer, 2018, S. 546–552.
- [13] Tuan D. Pham. „Classification of COVID-19 chest X-rays with deep learning: new models or fine tuning?“ In: *Health Information Science and Systems* 9.1 (22. Nov. 2020), S. 2. doi: [10.1007/s13755-020-00135-3](https://doi.org/10.1007/s13755-020-00135-3).
- [14] Adam Paszke u. a. „PyTorch: An imperative style, high-performance deep learning library“. In: *Advances in neural information processing systems* 32. Hrsg. von H. Wallach u. a. Curran Associates, Inc., 2019, S. 8024–8035.
- [15] *CheXpert: A Large Dataset of Chest X-Rays and Competition for Automated Chest X-Ray Interpretation*. URL: <https://stanfordmlgroup.github.io/competitions/chexpert/> (besucht am 08.10.2021).
- [16] Ivo M. Baltruschat u. a. „When does bone suppression and lung field segmentation improve chest x-ray disease classification?“ In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019, S. 1362–1366.
- [17] 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. Issue: 01. 2019, S. 590–597.
- [18] Qure.ai | AI for Chest X-rays. URL: <https://qure.ai/qxr.html> (besucht am 08.10.2021).
- [19] Qure.ai. *Qure.ai's qXR Becomes First AI-based Chest X-ray Interpretation Tool to Receive CE Certification*. URL: <https://www.prnewswire.com/in/news-releases/qureais-qxr-becomes-first-ai-based-chest-x-ray-interpretation-tool-to-receive-ce-certification-684141861.html> (besucht am 08.10.2021).
- [20] Preetham Putha u. a. „Can Artificial Intelligence Reliably Report Chest X-Rays?: Radiologist Validation of an Algorithm trained on 2.3 Million X-Rays“. In: *arXiv:1807.07455 [cs]* (4. Juni 2019). arXiv: [1807.07455](https://arxiv.org/abs/1807.07455).
- [21] Olga Russakovsky u. a. „ImageNet large scale visual recognition challenge“. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), S. 211–252. doi: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y).
- [22] Lunit Inc. URL: <https://www.lunit.io/en/products/insight-cxr> (besucht am 08.10.2021).
- [23] Lunit. *Fujifilm Introduces its AI-powered Product for Chest X-ray in Japan, in Collaboration with Lunit*. URL: <https://www.prnewswire.com/news-releases/fujifilm-introduces-its-ai-powered-product-for-chest-x-ray-in-japan-in-collaboration-with-lunit-301354317.html> (besucht am 08.10.2021).

- [24] Connor Shorten und Taghi M. Khoshgoftaar. „A survey on Image Data Augmentation for Deep Learning“. In: *Journal of Big Data* 6.1 (6. Juli 2019), S. 60. doi: [10.1186/s40537-019-0197-0](https://doi.org/10.1186/s40537-019-0197-0).
- [25] *Torchvision documentation*. URL: <https://pytorch.org/vision/stable/index.html> (besucht am 08.10.2021).
- [26] Aston Zhang u. a. *Dive into Deep Learning*. 2020.
- [27] Erkan Deniz u. a. „Transfer learning based histopathologic image classification for breast cancer detection“. In: *Health Information Science and Systems* 6.1 (28. Sep. 2018). doi: [10.1007/s13755-018-0057-x](https://doi.org/10.1007/s13755-018-0057-x).
- [28] Sergey Ioffe und Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *arXiv:1502.03167 [cs]* (2. März 2015). arXiv: [1502.03167](https://arxiv.org/abs/1502.03167).
- [29] 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.
- [30] Gao Huang u. a. „Densely connected convolutional networks“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 4700–4708.
- [31] Alok Kumar und Mayank Jain. *Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases*. Apress, 2020. doi: [10.1007/978-1-4842-5940-5](https://doi.org/10.1007/978-1-4842-5940-5).
- [32] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep learning*. MIT Press, 2016.
- [33] Hans-Joachim Mittag. *Statistik: Eine Einführung mit interaktiven Elementen*. 3. Aufl. Springer-Lehrbuch. Springer Spektrum, 2014. doi: [10.1007/978-3-642-54387-6](https://doi.org/10.1007/978-3-642-54387-6).
- [34] Dr. Fabian Brunner. „Ensemble-lernen“. Amberg, 8. Dez. 2020.
- [35] Tianqi Chen u. a. „Xgboost: extreme gradient boosting“. In: *R package version 0.4-2 1.4* (2015), S. 1–4.
- [36] Simon Haykin. *Neural Networks: A Comprehensive Foundation: A Comprehensive Foundation: United States Edition*. Subsequent Edition. Upper Saddle River, N.J.: Pearson, 1. Aug. 1998. 842 S.
- [37] Sebastian Raschka. *Machine learning mit Python und Scikit-learn und TensorFlow*. 2., aktualisierte und erweiterte Auflage. Frechen: mitp, 2018. 577 Seiten.
- [38] Rich Caruana u. a. „Ensemble selection from libraries of models“. In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, S. 18.
- [39] Eli Stevens, Luca Antiga und Thomas Viehmann. *Deep Learning with PyTorch*. 1. Edition. Manning Publications, 9. Juni 2020. 450 S.
- [40] David Powers. „Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation“. In: *Mach. Learn. Technol.* 2 (1. Jan. 2008).

- [41] Charles E. Metz. „Basic principles of ROC analysis“. In: *Seminars in Nuclear Medicine* 8.4 (Okt. 1978), S. 283–298. doi: [10.1016/S0001-2998\(78\)80014-2](https://doi.org/10.1016/S0001-2998(78)80014-2).
- [42] Joseph Paul Cohen. *ieee8023/covid-chestxray-dataset*. 1. Juni 2021.
- [43] Tom Fawcett. „ROC Graphs: Notes and Practical Considerations for Data Mining Researchers“. In: (), S. 28.
- [44] Peter A. Flach, José Hernández-Orallo und Cèsar Ferri Ramirez. „A coherent interpretation of AUC as a measure of aggregated classification performance“. In: *ICML*. 2011.
- [45] Grigorios Tsoumakas und Ioannis Vlahavas. „Random k-Labelsets: An Ensemble Method for Multilabel Classification“. In: *Machine Learning: ECML 2007*. Hrsg. von Joost N. Kok u. a. Bd. 4701. ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 406–417. doi: [10.1007/978-3-540-74958-5\\_38](https://doi.org/10.1007/978-3-540-74958-5_38).
- [46] Eneldo Loza Mencía. „Efficient Pairwise Multilabel Classification“. Dissertation. Darmstadt: Technische Universität, 4. Juli 2013.
- [47] Francisco Charte u. a. „MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation“. In: *Knowledge-Based Systems* 89 (1. Nov. 2015), S. 385–397. doi: [10.1016/j.knosys.2015.07.019](https://doi.org/10.1016/j.knosys.2015.07.019).
- [48] Jesse Read u. a. „Classifier Chains for Multi-label Classification“. In: *Machine Learning and Knowledge Discovery in Databases*. Hrsg. von Wray Buntine u. a. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, S. 254–269. doi: [10.1007/978-3-642-04174-7\\_17](https://doi.org/10.1007/978-3-642-04174-7_17).
- [49] Bulent Siyah. *ImageNet Winning CNN Architectures (ILSVRC) | Data Science and Machine Learning*. kaggle.com. 8. Mai 2020. URL: [a](#) (besucht am 08.10.2021).
- [50] 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.
- [51] Christiane Fellbaum. „WordNet: An electronic lexical database and some of its applications“. In: (1998). Publisher: MIT press Cambridge.
- [52] I. Allaouzi und M. Ben Ahmed. „A Novel Approach for Multi-Label Chest X-Ray Classification of Common Thorax Diseases“. In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, S. 64279–64288. doi: [10.1109/ACCESS.2019.2916849](https://doi.org/10.1109/ACCESS.2019.2916849).
- [53] Kaiming He, Ross Girshick und Piotr Dollár. „Rethinking imagenet pre-training“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, S. 4918–4927.
- [54] Tsung-Yi Lin u. a. „Microsoft COCO: Common Objects in Context“. In: *arXiv:1405.0312 [cs]* (20. Feb. 2015). arXiv: [1405.0312](https://arxiv.org/abs/1405.0312).
- [55] Simon Kornblith, Jonathon Shlens und Quoc V. Le. „Do better imagenet models transfer better?“ In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, S. 2661–2671.
- [56] Maithra Raghu u. a. „Transfusion: Understanding transfer learning for medical imaging“. In: *arXiv preprint arXiv:1902.07208* (2019).

- [57] Varun Gulshan u. a. „Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs“. In: *Jama* 316.22 (2016). Publisher: American Medical Association, S. 2402–2410.
- [58] Christian Szegedy u. a. „Rethinking the inception architecture for computer vision“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 2818–2826.
- [59] RSNA Pneumonia Detection Challenge. URL: <https://kaggle.com/c/rsna-pneumonia-detection-challenge> (besucht am 08.10.2021).
- [60] 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.
- [61] NIH Chest X-rays. URL: <https://kaggle.com/nih-chest-xrays/data> (besucht am 08.10.2021).
- [62] Alistair E. W. Johnson u. a. „MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports“. In: *Scientific Data* 6.1 (12. Dez. 2019). Number: 1 Publisher: Nature Publishing Group, S. 317. DOI: [10.1038/s41597-019-0322-0](https://doi.org/10.1038/s41597-019-0322-0).
- [63] Yifan Peng u. a. „NegBio: a high-performance tool for negation and uncertainty detection in radiology reports“. In: *AMIA Summits on Translational Science Proceedings 2018* (2018). Publisher: American Medical Informatics Association, S. 188.
- [64] Aurelia Bustos u. a. „PadChest: A large chest x-ray image dataset with multi-label annotated reports“. In: *Medical Image Analysis* 66 (1. Dez. 2020), S. 101797. DOI: [10.1016/j.media.2020.101797](https://doi.org/10.1016/j.media.2020.101797).
- [65] Alistair EW Johnson u. a. „MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs“. In: *arXiv preprint arXiv:1901.07042* (2019).
- [66] Albert L. Baert, Hrsg. *Encyclopedia of Diagnostic Imaging*. Berlin Heidelberg: Springer-Verlag, 2008.
- [67] Hina Amin und Waqas J Siddiqui. „Cardiomegaly“. In: *StatPearls [internet]* (2020).
- [68] Craig Hacking. *Lobar consolidation | Radiology Reference Article | Radiopaedia.org*. Radiopaedia. URL: <https://radiopaedia.org/articles/lobar-consolidation> (besucht am 08.10.2021).
- [69] *Pulmonary edema - Symptoms and causes*. Mayo Clinic. URL: <https://www.mayoclinic.org/diseases-conditions/pulmonary-edema/symptoms-causes/syc-20377009> (besucht am 08.10.2021).
- [70] Yuranga Weerakkody. *Pulmonary edema | Radiology Reference Article | Radiopaedia.org*. Radiopaedia. URL: <https://radiopaedia.org/articles/pulmonary-oedema> (besucht am 08.10.2021).

- [71] Yu-Xing Tang u. a. „Automated abnormality classification of chest radiographs using deep convolutional neural networks“. In: *npj Digital Medicine* 3.1 (14. Mai 2020). Number: 1 Publisher: Nature Publishing Group, S. 1–8. doi: [10.1038/s41746-020-0273-z](https://doi.org/10.1038/s41746-020-0273-z).
- [72] Dina Demner-Fushman u. a. „Preparing a collection of radiology examinations for distribution and retrieval“. In: *Journal of the American Medical Informatics Association: JAMIA* 23.2 (März 2016), S. 304–310. doi: [10.1093/jamia/ocv080](https://doi.org/10.1093/jamia/ocv080).
- [73] Daniel Kermany, Kang Zhang und Michael Goldbaum. „Large dataset of labeled optical coherence tomography (oct) and chest x-ray images“. In: *Mendeley Data*, v3 <http://dx.doi.org/10.17632/rscbjr9sj3> (2018).
- [74] Jared A. Dunnmon u. a. „Assessment of convolutional neural networks for automated classification of chest radiographs“. In: *Radiology* 290.2 (2019). Publisher: Radiological Society of North America, S. 537–544.
- [75] M. Abdullah-Al-Wadud u. a. „A Dynamic Histogram Equalization for Image Contrast Enhancement“. In: *Consumer Electronics, IEEE Transactions on* 53 (1. Juni 2007), S. 593–600. doi: [10.1109/TCE.2007.381734](https://doi.org/10.1109/TCE.2007.381734).
- [76] Wenwu Ye u. a. „Weakly Supervised Lesion Localization With Probabilistic-CAM Pooling“. In: *arXiv:2005.14480 [cs]* (29. Mai 2020). arXiv: [2005.14480](https://arxiv.org/abs/2005.14480).
- [77] Bolei Zhou u. a. „Learning deep features for discriminative localization“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 2921–2929.
- [78] yww211. *jfhealthcare/Chexpert*. original-date: 2019-11-03T09:10:37Z. 16. Aug. 2021.
- [79] Feng Li u. a. „Improved detection of focal pneumonia by chest radiography with bone suppression imaging“. In: *European Radiology* 22.12 (1. Dez. 2012), S. 2729–2735. doi: [10.1007/s00330-012-2550-y](https://doi.org/10.1007/s00330-012-2550-y).
- [80] Jens von Berg u. a. „A novel bone suppression method that improves lung nodule detection“. In: *International journal of computer assisted radiology and surgery* 11.4 (2016). Publisher: Springer, S. 641–655.
- [81] Maxim Gusarev u. a. „Deep learning models for bone suppression in chest radiographs“. In: 1. Aug. 2017, S. 1–7. doi: [10.1109/CIBCB.2017.8058543](https://doi.org/10.1109/CIBCB.2017.8058543).
- [82] Tom Brosch und Axel Saalbach. „Foveal fully convolutional nets for multi-organ segmentation“. In: *Medical Imaging 2018: Image Processing*. Bd. 10574. International Society for Optics und Photonics. 2018, 105740U.
- [83] Matthew Le u. a. „Neural Relational Autoregression for High-Resolution COVID-19 Forecasting“. In: (), S. 13.
- [84] Juan Carlos Quiroz u. a. „Development and Validation of a Machine Learning Approach for Automated Severity Assessment of COVID-19 Based on Clinical and Imaging Data: Retrospective Study“. In: *JMIR Medical Informatics* 9.2 (11. Feb. 2021), e24572. doi: [10.2196/24572](https://doi.org/10.2196/24572).
- [85] Duy M H Nguyen u. a. „An Attention Mechanism using Multiple Knowledge Sources for COVID-19 Detection from CT Images“. In: (), S. 9.

- [86] Asmaa Abbas, Mohammed M. Abdelsamea und Mohamed Medhat Gaber. „Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network“. In: *Applied Intelligence* 51.2 (1. Feb. 2021), S. 854–864. doi: [10.1007/s10489-020-01829-7](https://doi.org/10.1007/s10489-020-01829-7).
- [87] Shervin Minaee u. a. „Deep-COVID: Predicting COVID-19 from chest X-ray images using deep transfer learning“. In: *Medical Image Analysis* 65 (Okt. 2020), S. 101794. doi: [10.1016/j.media.2020.101794](https://doi.org/10.1016/j.media.2020.101794).
- [88] Subhankar Roy u. a. „Deep Learning for Classification and Localization of COVID-19 Markers in Point-of-Care Lung Ultrasound“. In: *IEEE Transactions on Medical Imaging* 39.8 (Aug. 2020), S. 2676–2687. doi: [10.1109/TMI.2020.2994459](https://doi.org/10.1109/TMI.2020.2994459).
- [89] Joseph Paul Cohen u. a. „COVID-19 Image Data Collection: Prospective Predictions Are the Future“. In: *arXiv:2006.11988 [cs, eess, q-bio]* (14. Dez. 2020). arXiv: [2006.11988](https://arxiv.org/abs/2006.11988).
- [90] José Daniel López-Cabrera u. a. „Current limitations to identify COVID-19 using artificial intelligence with chest X-ray imaging“. In: *Health and Technology* 11.2 (2021). Publisher: Springer, S. 411–424.
- [91] Erdi Çallı u. a. „FRODO: Free rejection of out-of-distribution samples: application to chest x-ray analysis“. In: *arXiv preprint arXiv:1907.01253* (2019).
- [92] Prasanta Chandra Mahalanobis. „On the generalized distance in statistics“. In: National Institute of Science of India, 1936.
- [93] R. De Maesschalck, D. Jouan-Rimbaud und D. L. Massart. „The Mahalanobis distance“. In: *Chemometrics and Intelligent Laboratory Systems* 50.1 (4. Jan. 2000), S. 1–18. doi: [10.1016/S0169-7439\(99\)00047-7](https://doi.org/10.1016/S0169-7439(99)00047-7).
- [94] Dan Hendrycks und Kevin Gimpel. „A baseline for detecting misclassified and out-of-distribution examples in neural networks“. In: *arXiv preprint arXiv:1610.02136* (2016).
- [95] Hieu H. Pham u. a. „Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels“. In: *arXiv:1911.06475 [cs, eess]* (12. Juni 2020). arXiv: [1911.06475](https://arxiv.org/abs/1911.06475).
- [96] Diederik P. Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: *arXiv:1412.6980 [cs]* (29. Jan. 2017).
- [97] Christian Szegedy u. a. „Inception-v4, inception-resnet and the impact of residual connections on learning“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Bd. 31. Issue: 1. 2017.
- [98] François Chollet. „Xception: Deep learning with depthwise separable convolutions“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 1251–1258.
- [99] Barret Zoph u. a. „Learning transferable architectures for scalable image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, S. 8697–8710.

- [100] Qingji Guan u. a. „Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification“. In: *arXiv preprint arXiv:1801.09927* (2018).
- [101] Yuncheng Li, Yale Song und Jiebo Luo. „Improving pairwise ranking for multi-label image classification“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 3617–3625.
- [102] Zhuoning Yuan u. a. „Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification“. In: *arXiv:2012.03173 [cs, math, stat]* (5. Dez. 2020). arXiv: [2012.03173](https://arxiv.org/abs/2012.03173).
- [103] Corinna Cortes und Mehryar Mohri. „AUC optimization vs. error rate minimization“. In: *Advances in neural information processing systems* 16.16 (2004). Publisher: MIT Press, S. 313–320.
- [104] Zhuoning Yuan. *yzhuoning/LibAUC*. original-date: 2021-03-31T02:51:18Z. 27. Juli 2021.
- [105] *nn package — PyTorch Tutorials 1.9.0+cu102 documentation*. URL: [https://pytorch.org/tutorials/beginner/former\\_torchies/nnft\\_tutorial.html](https://pytorch.org/tutorials/beginner/former_torchies/nnft_tutorial.html) (besucht am 08. 10. 2021).
- [106] *scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation*. URL: <https://scikit-learn.org/stable/index.html> (besucht am 08. 10. 2021).
- [107] Will Badr. *5 Ways to Detect Outliers That Every Data Scientist Should Know (Python Code)*. Medium. 12. Apr. 2019. URL: <https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623> (besucht am 08. 10. 2021).
- [108] *OpenCV: Template Matching*. URL: [https://docs.opencv.org/master/d4/dc6/tutorial\\_py\\_template\\_matching.html](https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html) (besucht am 08. 10. 2021).
- [109] Olaf Ronneberger, Philipp Fischer und Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation“. In: *arXiv:1505.04597 [cs]* (18. Mai 2015). arXiv: [1505.04597](https://arxiv.org/abs/1505.04597).
- [110] *Label image regions — skimage v0.19.0.dev0 docs*. URL: [https://scikit-image.org/docs/dev/auto\\_examples/segmentation/plot\\_label.html](https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_label.html) (besucht am 08. 10. 2021).
- [111] P. Szymański und T. Kajdanowicz. „A scikit-based Python environment for performing multi-label classification“. In: *ArXiv e-prints* (Feb. 2017). arXiv: [1702.01460\[cs.LG\]](https://arxiv.org/abs/1702.01460).
- [112] Wendy Navarrete. *PCA with numpy*. Medium. 6. Juni 2020. URL: <https://towardsdatascience.com/pca-with-numpy-58917c1d0391> (besucht am 08. 10. 2021).
- [113] Dmytro Danevskyi. *ex4sperans/freesound-classification*. original-date: 2019-04-12T18:15:06Z. 1. Juni 2021.
- [114] Stephen M. Pizer u. a. „Adaptive histogram equalization and its variations“. In: *Computer vision, graphics, and image processing* 39.3 (1987). Publisher: Elsevier, S. 355–368.

- [115] Yanmin Sun, Andrew KC Wong und Mohamed S. Kamel. „Classification of imbalanced data: A review“. In: *International journal of pattern recognition and artificial intelligence* 23.4 (2009). Publisher: World Scientific, S. 687–719.
- [116] Nitesh V. Chawla u. a. „SMOTE: synthetic minority over-sampling technique“. In: *Journal of artificial intelligence research* 16 (2002), S. 321–357.
- [117] *Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.9.0+cu102 documentation*. URL: [https://pytorch.org/tutorials/beginner/transfer\\_learning\\_tutorial.html](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html) (besucht am 08.10.2021).
- [118] *Automatic Mixed Precision package - torch.cuda.amp — PyTorch 1.9.0 documentation*. URL: <https://pytorch.org/docs/stable/amp.html> (besucht am 08.10.2021).
- [119] Mark Sandler u. a. „MobileNetV2: Inverted Residuals and Linear Bottlenecks“. In: *arXiv:1801.04381 [cs]* (21. März 2019). arXiv: [1801.04381](https://arxiv.org/abs/1801.04381).
- [120] Forrest N. Iandola u. a. „SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size“. In: *arXiv:1602.07360 [cs]* (4. Nov. 2016). arXiv: [1602.07360](https://arxiv.org/abs/1602.07360).
- [121] Saining Xie u. a. „Aggregated Residual Transformations for Deep Neural Networks“. In: *arXiv:1611.05431 [cs]* (10. Apr. 2017). arXiv: [1611.05431](https://arxiv.org/abs/1611.05431).
- [122] *Transfer learning and fine-tuning | TensorFlow Core*. TensorFlow. URL: [https://www.tensorflow.org/guide/keras/transfer\\_learning](https://www.tensorflow.org/guide/keras/transfer_learning) (besucht am 08.10.2021).
- [123] Xiangyu Zhang u. a. „ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices“. In: *arXiv:1707.01083 [cs]* (7. Dez. 2017). arXiv: [1707.01083](https://arxiv.org/abs/1707.01083).
- [124] Sergey Zagoruyko und Nikos Komodakis. „Wide Residual Networks“. In: *arXiv:1605.07146 [cs]* (14. Juni 2017). arXiv: [1605.07146](https://arxiv.org/abs/1605.07146).
- [125] *CodaLab Worksheets*. URL: <https://worksheets.codalab.org/home> (besucht am 08.10.2021).
- [126] *CheXpert Submission Tutorial*. URL: <https://worksheets.codalab.org/worksheets/0x693b0063ee504702b21f94ffb2d99c6d/> (besucht am 08.10.2021).
- [127] Ninareh Mehrabi u. a. „A survey on bias and fairness in machine learning“. In: *ACM Computing Surveys (CSUR)* 54.6 (2021). Publisher: ACM New York, NY, USA, S. 1–35.
- [128] Laleh Seyyed-Kalantari u. a. „CheXclusion: Fairness gaps in deep chest X-ray classifiers“. In: *Biocomputing 2021. Pacific Symposium on Biocomputing 2021*. Kohala Coast, Hawaii, USA: WORLD SCIENTIFIC, Nov. 2020, S. 232–243. doi: [10.1142/9789811232701\\_0022](https://doi.org/10.1142/9789811232701_0022).
- [129] *About Us | Medicaid*. URL: <https://www.medicaid.gov/about-us/index.html> (besucht am 08.10.2021).
- [130] Trishan Panch, Heather Mattie und Leo Anthony Celi. „The “inconvenient truth” about AI in healthcare“. In: *npj Digital Medicine* 2.1 (16. Aug. 2019). Number: 1. Publisher: Nature Publishing Group, S. 1–3. doi: [10.1038/s41746-019-0155-4](https://doi.org/10.1038/s41746-019-0155-4).

- [131] *Schlussbericht der Enquête-Kommission „Künstliche Intelligenz – Gesellschaftliche Verantwortung und wirtschaftliche, soziale und ökologische Potenziale“.* Schlussbericht Drucksache 19/23700. Deutscher Bundestag, 28. Okt. 2020, S. 806.

# Abbildungsverzeichnis

1.1	Darstellung des Aufbaus dieser Arbeit. . . . .	3
2.1	Vergleich der Aufnahmepositionen eines Thorax-Röntgenbildes. . . . .	5
2.2	Darstellung der Convolution Operation. . . . .	8
2.3	Beispiel für die Berechnung einer Convolution. . . . .	8
2.4	Vergleich des Trainings- und Testfehlers mit 20 respektive 56 Schichten .	12
2.5	Darstellung einer shortcut connection. . . . .	12
2.6	Vergleich zwischen ResNets und CNNs ohne shortcut connections. . . .	12
2.7	Darstellung der Blöcke für ResNets. . . . .	13
2.8	Übersicht über die ResNet-Architekturen. . . . .	13
2.9	Übersicht über die DenseNet-Architekturen. . . . .	14
2.10	Übersicht der verschiedenen Ensemble Möglichkeiten. . . . .	20
2.11	Plot der Sigmoid-Funktion. . . . .	24
2.12	Darstellung einer Konfusionsmatrix. . . . .	25
2.13	Beispiel einer Konfusionsmatrix . . . . .	26
2.14	Plot einer fiktiven ROC-Kurve. . . . .	29
2.15	Plot zweier fiktiver ROC-Kurven. . . . .	30
2.16	Visualisierung des Berechnungsprozesses für den ROC-AUC Wert. . . .	31
5.1	Anzahl der verfügbaren Thorax-Röntgenbilder und der veröffentlichten Arbeiten mit Bezug dazu. . . . .	44
5.2	Chord-Diagramm des ChestX-ray14-Datensatzes. . . . .	45
5.3	Chord-Diagramm für den CheXpert-Datensatz mit U-Ones Vorgehen. .	49
6.1	Übersicht über die Zuordnung der in [10] untersuchten Veröffentlichungen.	52
7.1	Pathologie und zugehöriges Prioritätslevel, wie sie in [11] verwendet wurden. . . . .	61
7.2	Hierarchie zwischen den Labeln des CheXpert-Datensatzes. . . . .	66
7.3	Architektur des Attention-Guided CNN. . . . .	69
7.4	Resultat des Vergleichs der Trainingsstrategie für das AG-CNN. . . . .	71
7.5	Resultat des Vergleichs der Strategie zur Erzeugung der Heatmaps aus den Aktivierungen. . . . .	71
7.6	Vergleich der gemittelten AUC-Werte für den Local und Fusion Branch für verschiedene Schwellwerte $\tau$ auf den Testdaten. . . . .	72
7.7	Vergleich zwischen den gelabelten Bounding-Boxes und den AG-CNN ROIs . . . . .	73
7.8	Darstellung der Kaskadierung aus [12]. . . . .	76

7.9	PESG-Algorithmus von Yuan et al. [102]. . . . .	78
7.10	Beispiel Intra-Epoch Evaluation. . . . .	80
8.1	Die ersten 16 als OoD erkannte Bilder des CheXpert-Datensatzes. . . . .	85
8.2	Vergleich dreier Template Matching Algorithmen. . . . .	87
8.3	Kumulative erklärte Varianz der $n$ Hauptkomponenten der Eigenzerlegung. . . . .	90
8.4	Beispiel für Histogram Equalization . . . . .	93
9.1	Klassenverteilung je Vergleichslabel nach dem Ausbalancieren. . . . .	98
9.2	Verteilung der Klassen je Label aller Label nach dem Ausbalancieren der Vergleichslabel. Der Datensatz weist insgesamt weiterhin starke Imbalance auf. . . . .	99
9.3	Klassenverteilung je Vergleichslabel nach dem Angleichen an die Validierungsdaten. . . . .	100
9.4	Loss-Werte mit den Strategien Mean Var init und weight transfusion. . .	105
10.2	ROC-AUC-Kurven des gewählten Ensembles. . . . .	123
B.1	Ausgangsbild für die Bildaugmentierungen aus dem CheXpert [17] Datensatz. . . . .	156
B.2	Zufälliges Horizontales Spiegeln mit einer Wahrscheinlichkeit von $p=0.5$	157
B.3	Zufälliges Rotieren im Bereich von $\pm 25$ Grad. . . . .	158
B.4	Zufälliges Ausschneiden eines Bildbereiches mit 224x224 Pixeln . . . . .	159
B.5	Normalisierung mit Mittelwert und Standardabweichung des ImageNet Datensatzes. . . . .	160
B.6	Chord-Diagramm für den CheXpert Datensatz mit U-Zeros Vorgehen. .	161
B.7	Verwendetes Template. . . . .	162
B.8	Vergleich der Laufzeit der naiven und Broadcasting PWE-Loss-Implementierung. . . . .	162
B.9	Klassenverteilung je Label der CheXpert Trainingsdaten . . . . .	163
B.10	Chord-Diagramm nach einer Iteration des MLSMOTE-Vorgehens. . .	164
B.11	Verteilung der Klassen je Label bei U-Ones Vorgehen. . . . .	165
B.12	Klassenverteilung je Label nach Oversampling. . . . .	166
B.13	Verteilung der Klassen je Label für die fünf Vergleichslabel. . . . .	167
B.14	Beispiel Intra-Epoch Evaluation. . . . .	167
B.15	Beispiel Intra-Epoch Evaluation. . . . .	168
B.16	Verlauf der Loss-Werte für ein MobileNet V2 mit auf dem ImageNet Datensatz vortrainierten Gewichten . . . . .	169
B.17	Korrelation der Ensemble-Kandidaten für das Label Atelectasis . . . .	170
B.18	Korrelation der Ensemble-Kandidaten für das Label Cardiomegaly . .	171
B.19	Korrelation der Ensemble-Kandidaten für das Label Consolidation . .	172
B.20	Korrelation der Ensemble-Kandidaten für das Label Edema . . . . .	173
B.21	Korrelation der Ensemble-Kandidaten für das Label Pleural Effusion .	174

# Tabellenverzeichnis

5.1	Verteilung der Klassen innerhalb der Label des CheXpert-Datensatzes.. .	47
7.1	Vergleich der AG-CNN Branches für ResNet-50 und DenseNet-121 Backbones aus [100] .. . . . .	70
8.1	Ergebnisse der OoD-Detektion mit FRODO. . . . .	84
8.2	Klassenverteilung der Bilder, die als OoD erkannt wurden. . . . .	84
8.3	Vergleich der erklärten Varianz je nach Anzahl der gewählten Hauptkomponenten. . . . .	91
9.1	Ausgangssituation und daraus resultierende Maßnahme für das Ausbalancieren der Vergleichslabel. . . . .	98
9.2	Ausgangssituation und Maßnahme für das Angleichen der Vergleichslabel an den Validierungsdatensatz. . . . .	100
9.3	Resultate des Vergleichs der Maßnahmen. . . . .	104
9.4	Vergleich der Gewichtungen des Fusion Branch je Label. . . . .	106
9.5	Resultate der Experimente zur Optimierung der Methode aus [95]. . . . .	108
9.6	Resultate der Experimente zu [13]. . . . .	109
9.7	Resultate der weiteren Experimente zu [13]. . . . .	110
9.8	Resultate der Experimente zu [52]. . . . .	111
9.9	Resultate der Experimente für das Snapshot Ensemble. . . . .	114
9.10	Resultate aus den Experimenten zu [12]. . . . .	115
9.11	Resultate der Experimente für [102]. . . . .	116
10.1	Beste AUC-Werte eines einzelnen Modells für die einzelnen Pathologien.	119
10.2	Vergleich der AUC-Werte auf den Validierungsdaten mit einem Ensemble das $n$ Modelle enthält. . . . .	120
10.3	AUC Werte für die Stacking-Klassifizierer. . . . .	124
10.4	AUC Werte des Ensembles für die einzelnen Pathologien. . . . .	124
12.1	AUC-Werte des finalen Ensembles auf den Testdaten des ChestX-ray14-Datensates . . . . .	127
12.2	Vergleich der Evaluierung auf dem ChestX-ray14-Datensatz. . . . .	128
13.1	Auszug der Rangliste der CheXpert Competition. . . . .	131

# Listings

A.1	Minimalbeispiel eines Convolutional Neural Networks . . . . .	154
C.1	Beispielcode zum Speichern der Aktivierungen . . . . .	175
C.2	FRODO-Klasse zur OoD-Detektion . . . . .	175
C.3	Abgewandelte forward()-Methode der DenseNet-Klasse. . . . .	176
C.4	Funktionen zur Erzeugung und Skalierung der Heatmaps. . . . .	176
C.5	Erzeugen der Masken aus den Heatmaps. . . . .	177
C.6	Ausschneiden der quadratischen ROI aus dem Bild. . . . .	177
C.7	Identitäts-Schicht, die die Eingabe unverändert zurückgibt. . . . .	178
C.8	Klasse zur Erzeugung eines Kaskaden-Objektes eines beliebigen Levels.	178
C.9	Naive Implementierung des PWE Loss. . . . .	179
C.10	Broadcasting Implementierung des PWE Loss aus [113]. . . . .	179

## Anhang A

# Weitere Informationen

### A.1 Pytorch

Pytorch [14] ist eine Python-Bibliothek, die es ermöglicht, Deep Learning Projekte zu implementieren. Neben ihr existieren mehrere weitere ähnliche Bibliotheken, von denen besonders Tensorflow zu erwähnen ist. Die meisten Deep Learning Projekte verwenden eine dieser beiden, wobei sich in Unternehmen eine Tendenz zu Tensorflow und im Bereich der Forschung zu Pytorch erkennen lässt [39, S. 9]. Eine neuere Entwicklung ist JAX, das so wie Tensorflow von Google entwickelt wird und zunehmend an Verbreitung gewinnt [39, S. 9]. Jede dieser Bibliotheken bietet im Kern zwei Features, die das trainieren von Deep Learning Modellen ermöglichen: Die Möglichkeit, Berechnungen auf der Grafikkarte (GPU) durchzuführen und somit von Geschwindigkeitsvorteilen zu profitieren sowie zur Nutzung von Methoden zur numerischen Optimierung [39, S. 8], meistens anhand von Gradienten.

Für diese Arbeit fiel die Wahl aus persönlicher Präferenz und da viele Referenzveröffentlichungen damit umgesetzt wurden auf Pytorch. Die Implementierung wären aber auch in einem der anderen Frameworks möglich gewesen.

Im Folgenden werden die wichtigsten Grundkomponenten von Pytorch kurz vorgestellt.

#### A.1.1 Tensoren

Die grundlegende Datenstruktur in Pytorch heißt `torch.Tensor`. Tensoren sind allgemein multidimensionale Listen mit beliebiger Achsenanzahl [26, S. 57]. Im Folgenden sind Pytorch-Tensoren gemeint, wenn von Tensoren gesprochen wird.

Während die einzelnen Elemente einer normalem Python-Liste im Speicher an unterschiedlichen Stellen liegen, belegt ein Pytorch-Tensor einen zusammenhängenden Speicherbereich [39, S. 43]. Genauer gesagt, bildet der Tensor nur eine View auf den Speicherbereich, was dazu führt, dass der selbe Speicher von mehreren Tensoren auf

unterschiedliche Arten indiziert werden kann, ohne, dass der Speicher dafür kopiert werden müsste [39, S. 53f.].

Ein Tensor kann sowohl ganzzahlige Werte (`int`), Gleitkommazahlen (`float`) als auch boolsche Wahrheitswerte (`bool`) enthalten [39, S. 50]. Außerdem können Tensoren sowohl auf der GPU als auch auf dem Prozessor (CPU) existieren und mittels der Methode `tensor.to()` zwischen den beiden transferiert werden [39, S. 63].

### A.1.2 Autograd

Einer der essenziellen Schritte beim Trainieren eines Neuronalen Netzes ist das Aktualisieren der Parameter. Dies geschieht üblicherweise anhand einer Form des Gradientenabstiegsverfahrens (GAV) unter Verwendung der Kettenregel, beispielsweise mittels „Stochastic Gradient Descent“ (SGD). Dafür ist es nötig, den Gradienten, also den Vektor aller partiellen Ableitungen [26, S. 70], zu kennen.

In Pytorch geschieht die Berechnung der Gradienten automatisch über das Modul `autograd` [39, S. 123f.]. Man kann Pytorch entweder auf Ebene der Tensoren über `requires_grad=True` oder für einen Codeabschnitt mittels der Umgebung `with torch.set_grad_enabled(True)` mitteilen, dass der Berechnungsgraph aufgezeichnet werden soll. Möchte man das Gegenteil erreichen, zum Beispiel um die Verlustfunktion auf den Validierungsdaten auszuwerten, lässt sich die Aufzeichnung des Berechnungsgraphen mit `with torch.no_grad()` deaktivieren.

Dieser Graph, der bei jedem Forward-Pass durch das Modell erzeugt wird, kann dann für die Backpropagation mittels `loss.backward()` genutzt werden.

Dabei sollte man berücksichtigen, dass mit jedem Aufruf von `loss.backward()` die Gradienten an den Blattknoten des Graphen akkumuliert werden und man diese daher vor einem erneuten `loss.backward()` auf Null zurücksetzen muss [39, S. 125].

### A.1.3 Optimisierer

Nachdem die Gradienten zur Verfügung stehen ist es möglich, die Parameter so anzupassen, dass sie den Modellfehler (hoffentlich) reduzieren. Es gibt mehrere Algorithmen, die die genaue Art des Updates unterschiedlich realisieren. In dem Modul `torch.optim` wurden die häufigsten davon implementiert. Unter anderem lassen sich Klassen für den Stochastic Gradient Descent oder den Adam-Optimisierer finden.

Wenn man einen solchen Optimisierer initialisiert, übergibt man die Modellparameter, die dann bei Aufruf der Methode `optimizer.step()` basierend auf den Gradienten aktualisiert werden [39, S. 127f.]. Das eben angesprochene Nullsetzen der sich akkumulierenden Gradienten kann über die Methode `optimizer.zero_grad()` erreicht werden [39, S. 128].

### A.1.4 nn.Module

Die Basisklasse um Neuronale Netze zu implementieren heißt in Pytorch `torch.nn.Module`. Was meistens als *Schicht* beziehungsweise engl. *Layer* eines Neuronalen Netzes

bezeichnet wird, wird in Pytorch zusätzlich „module“ genannt [39, S. 151]. In `torch.nn` sind bereits mehrere typische Schichten implementiert, unter anderem vollverknüpfte (`torch.nn.Linear`), Convolutional (z.B. `torch.nn.Conv2d`) und Pooling Schichten (z.B. `torch.nn.MaxPool2d`, `torch.nn.AvgPool2d`), die alle für die Bildklassifikation verwendet werden. Möchte man eine eigene Schicht implementieren, sollte man die entsprechende Klasse von `torch.nn.Module` ableiten. Das erfordert die Implementierung der `forward()` Methode [39, S. 207]. Außerdem sollte man im Konstruktor, also der `__init__()` Methode alle nötigen Submodule als Member-Variablen instanziieren [39, S. 207].

Ein Minimalbeispiel eines Neuronalen Netzes mit einer Convolutional und Pooling sowie einer Linear Schicht wird in Listing A.1 gezeigt.

Obwohl die Methode `forward()` einen Forward-Pass durchführt, sollte man sie nicht direkt verwenden. Stattdessen sollte man die `__call__()` Methode dafür nutzen, statt `model.forward(x)` also `model(x)`. Dies liegt darin begründet, dass bei letzterer Funktion zusätzliche Funktionen aufgerufen werden bevor und nachdem die `forward()` Methode aufgerufen wird, beispielsweise sogenannte Hooks [39, S. 152].

```

1 from torch import nn
2 class CustomCNN(nn.Module):
3     def __init__(self):
4         super().__init__()
5         self.conv = nn.Conv2d(3, 8, kernel_size=(3,3), padding=1)
6         self.relu = nn.ReLU()
7         self.pool = nn.MaxPool2d(kernel_size=(2,2))
8         self.fc = nn.Linear(8*16*16, 1)
9
10    def forward(self, X):
11        # let b := batchsize
12        # input shape: b x 3 x 32 x 32
13
14        out = self.conv(X) # shape: b x 8 x 32 x 32
15        out = self.relu(out)
16        out = self.pool(out) # shape: b x 8 x 16 x 16
17        out = out.view(-1, 8*16*16) # flatten to b x (8*16*16)
18        out = self.fc(out) # shape: batchsize x 1
19
20    return out

```

**Listing A.1:** Minimalbeispiel eines Convolutional Neural Networks in Pytorch für Inputs der Dimension  $bx3x32x32$  und mit einer Ausgabeklasse

### A.1.5 Aktivierungs- und Lossfunktionen

Ein weiterer Grundbaustein für das Trainieren Neuronaler Netze sind Aktivierungs- und Lossfunktionen. Auch diese finden sich unter `torch.nn`. Es sind eine Vielzahl an Funktionen implementiert, darunter als Aktivierungsfunktionen die ReLU (`torch.nn.ReLU`), Sigmoid (`torch.nn.Sigmoid`) und der Tangens Hyperbolicus (`torch.nn.Tanh`) sowie als Lossfunktionen der BCE Loss (`torch.nn.BCELoss`) und der Negative Log Likelihood Loss (`torch.nn.NLLLoss`).

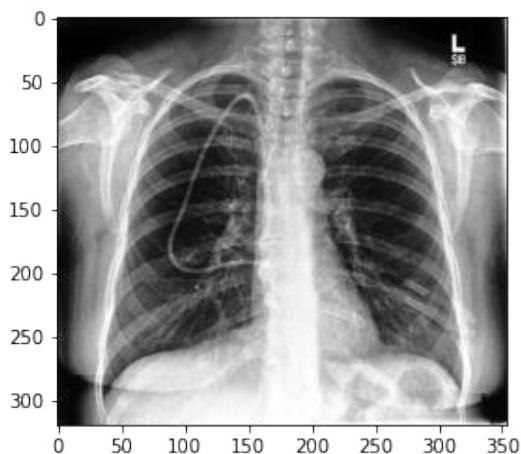
### A.1.6 Torchvision

Torchvision ist eine Bibliothek innerhalb des Pytorch-Projektes [25]. Sie besteht im Wesentlichen aus Implementierungen von erfolgreichen Netzwerkarchitekturen wie ResNet und DenseNet sowie der Möglichkeit, diese einfach mit vortrainierten Gewichten zu instanziieren, einigen Standard-Datensätzen wie CIFAR10 sowie Bildtransformationen beispielsweise für die Bildaugmentierung [25].

Damit ist sie neben Pytorch selbst eine der wichtigsten Bibliotheken für diese Arbeit.

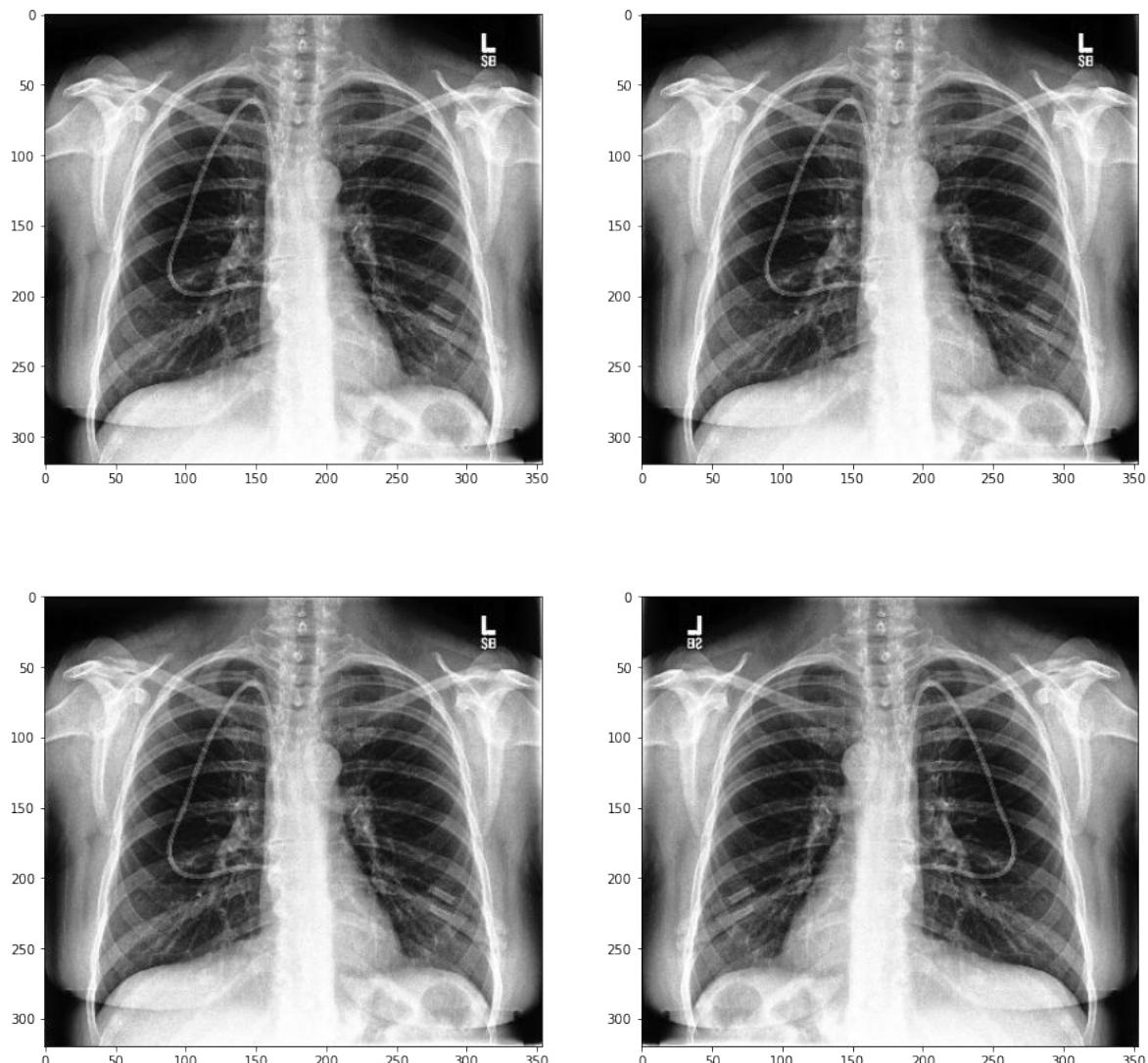
## Anhang B

### Abbildungen



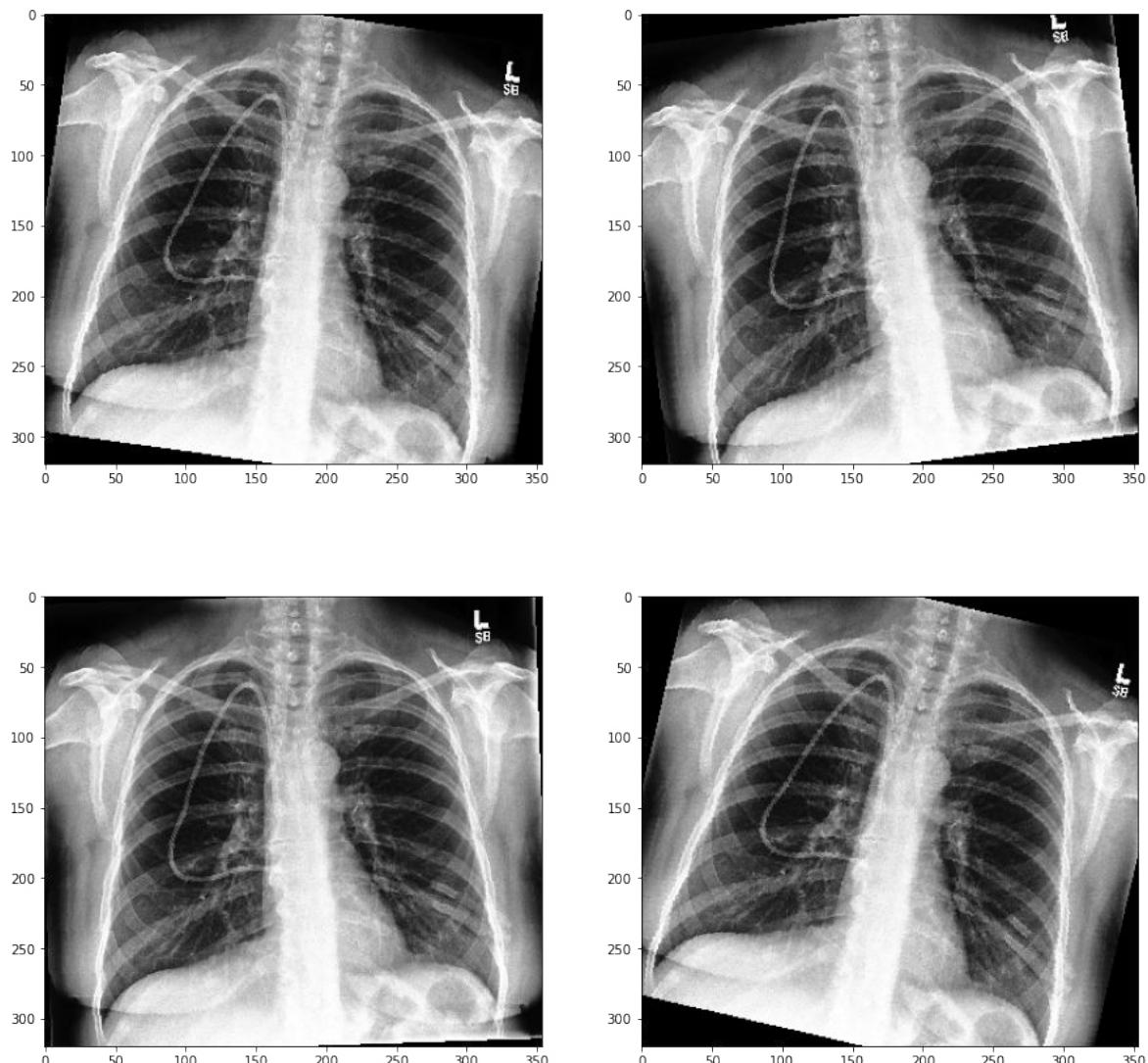
**Abbildung B.1:** Ausgangsbild für die Bildaugmentierungen aus dem CheXpert [17] Datensatz.

## Augmentierung: RandomHorizontalFlip



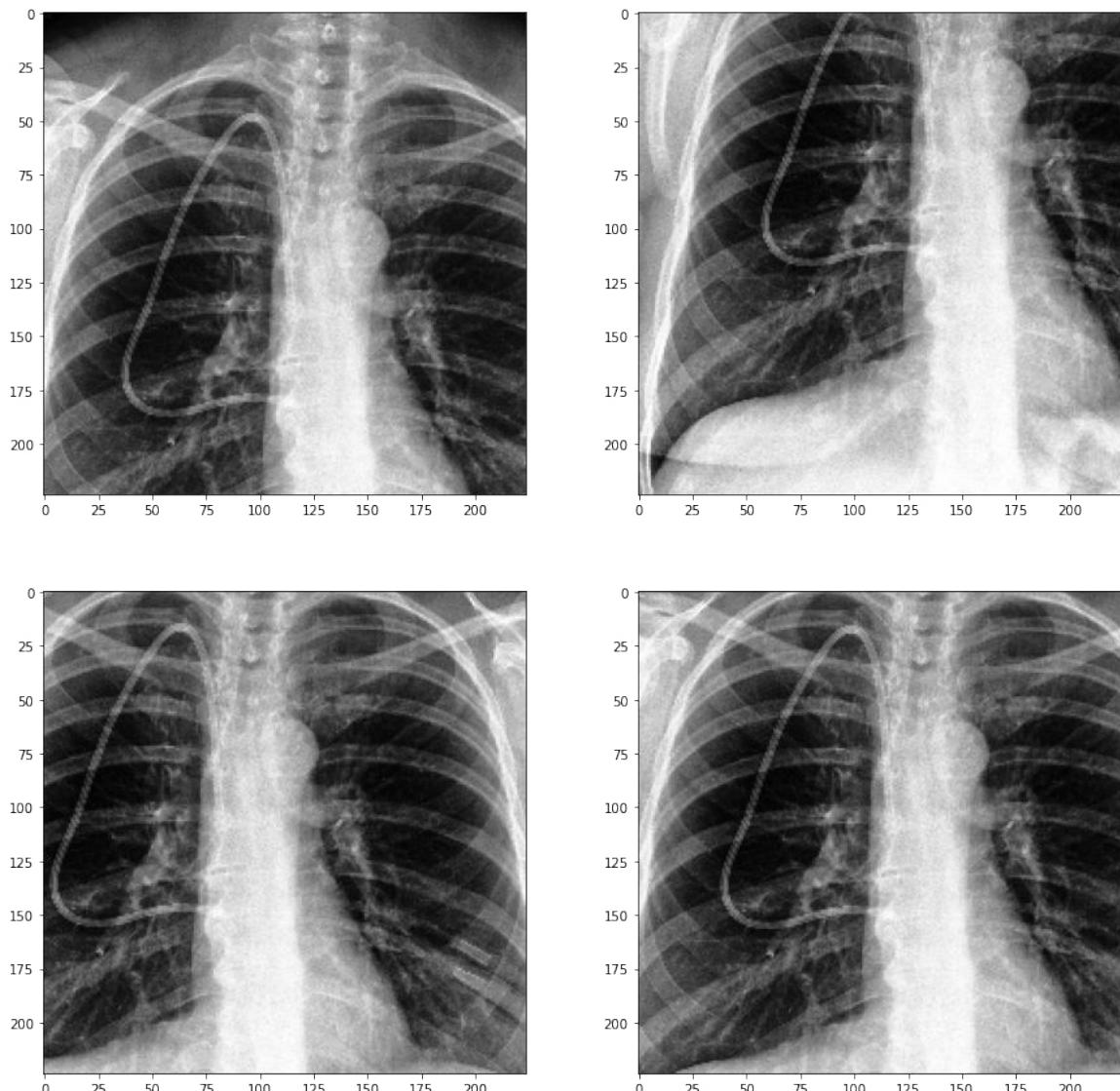
**Abbildung B.2:** Zufälliges Horizontales Spiegeln mit einer Wahrscheinlichkeit von  $p=0.5$

## Augmentierung: RandomRotation

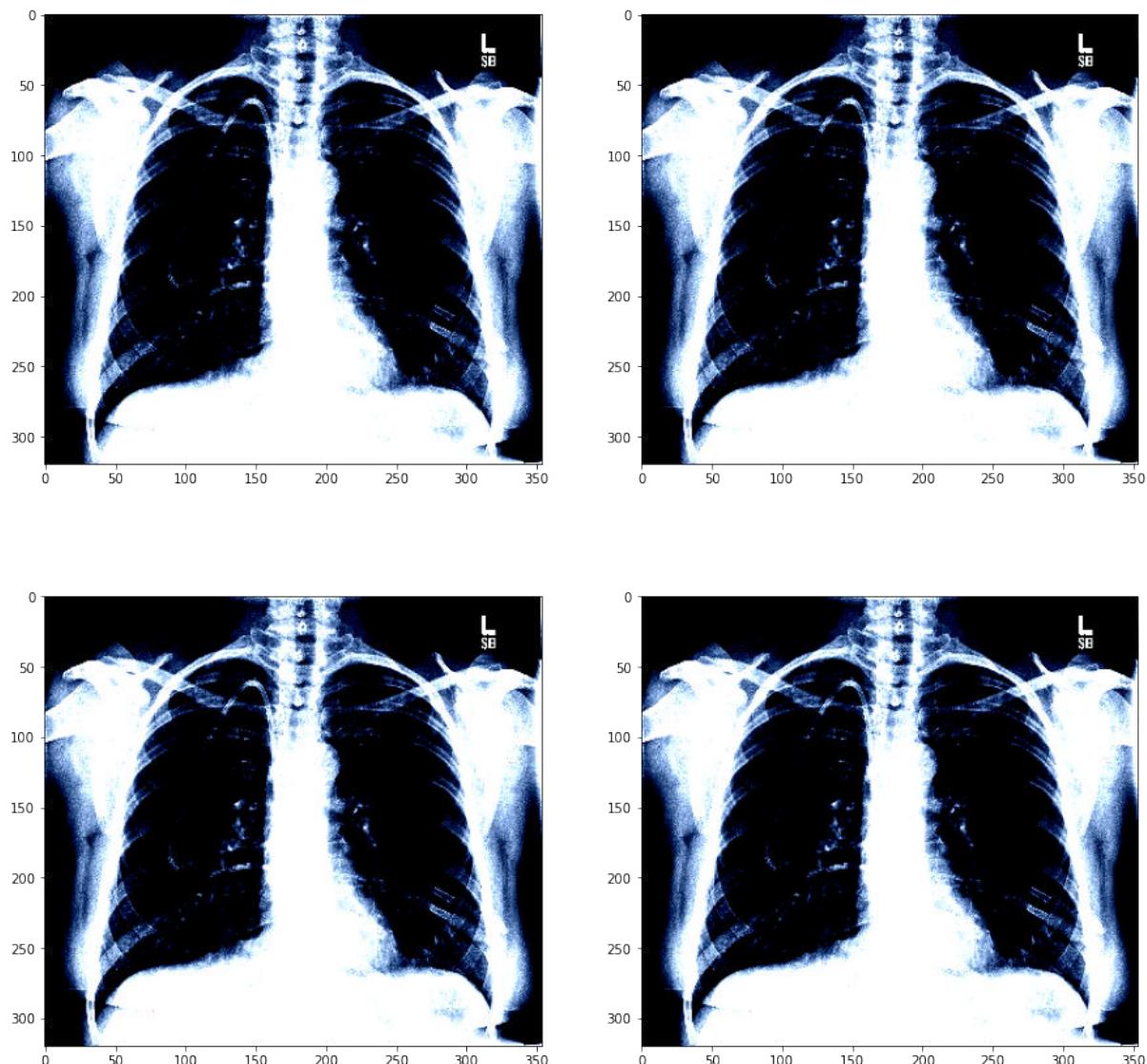


**Abbildung B.3:** Zufälliges Rotieren im Bereich von  $\pm 25$  Grad.

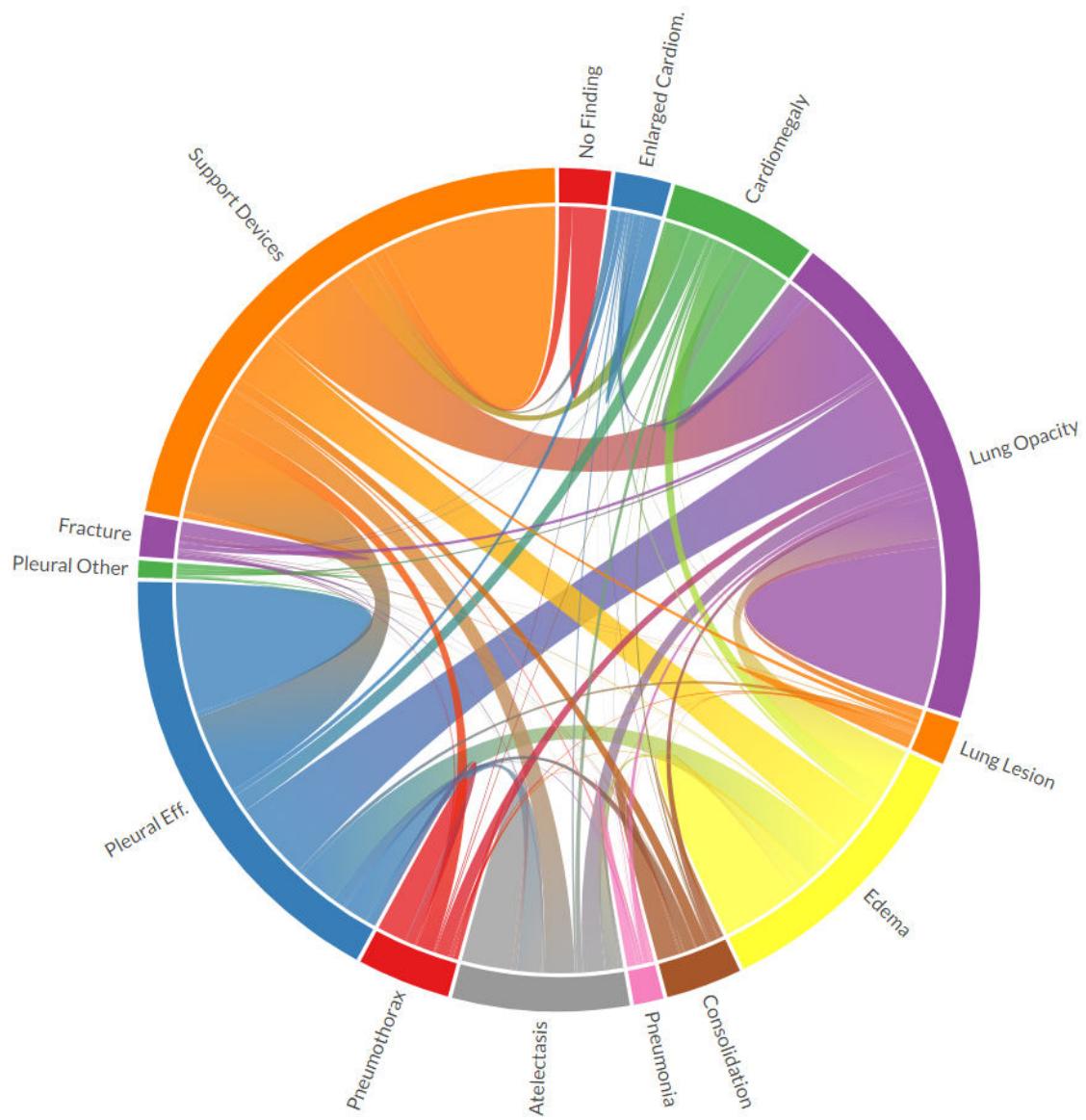
Augmentierung: RandomCrop

**Abbildung B.4:** Zufälliges Ausschneiden eines Bildbereiches mit 224x224 Pixeln

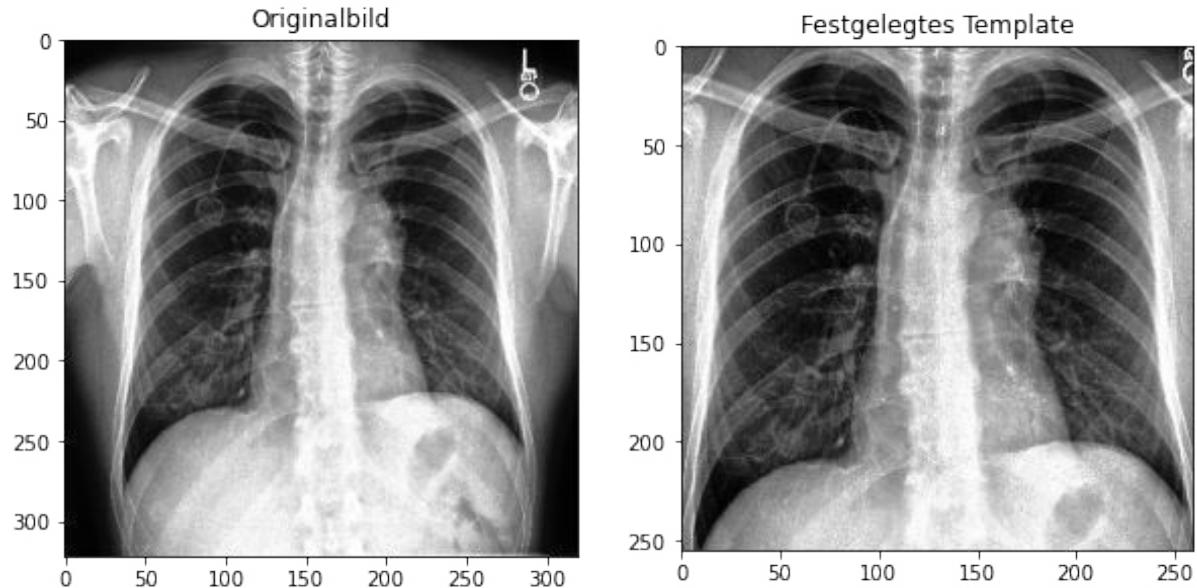
Augmentierung: Grayscale - ToTensor - Normalize



**Abbildung B.5:** Normalisierung mit Mittelwert und Standardabweichung des ImageNet Datensatzes. Hierbei gibt es kein probabilistisches Element, sondern die Augmentierung trägt zur Konvergenz des Trainings bei.



**Abbildung B.6:** Chord-Diagramm für den CheXpert-Datensatz mit U-Zeros Vorgehen. Die interaktive Version befindet sich auf dem beiliegenden Datenträger.



**Abbildung B.7:** Originalbild (links) sowie daraus erzeugtes Template (rechts) für das Template Matching. Das Originalbild stammt aus dem CheXpert-Datensatz, Patient 5, Studie 1, view1\_frontal.jpg

```

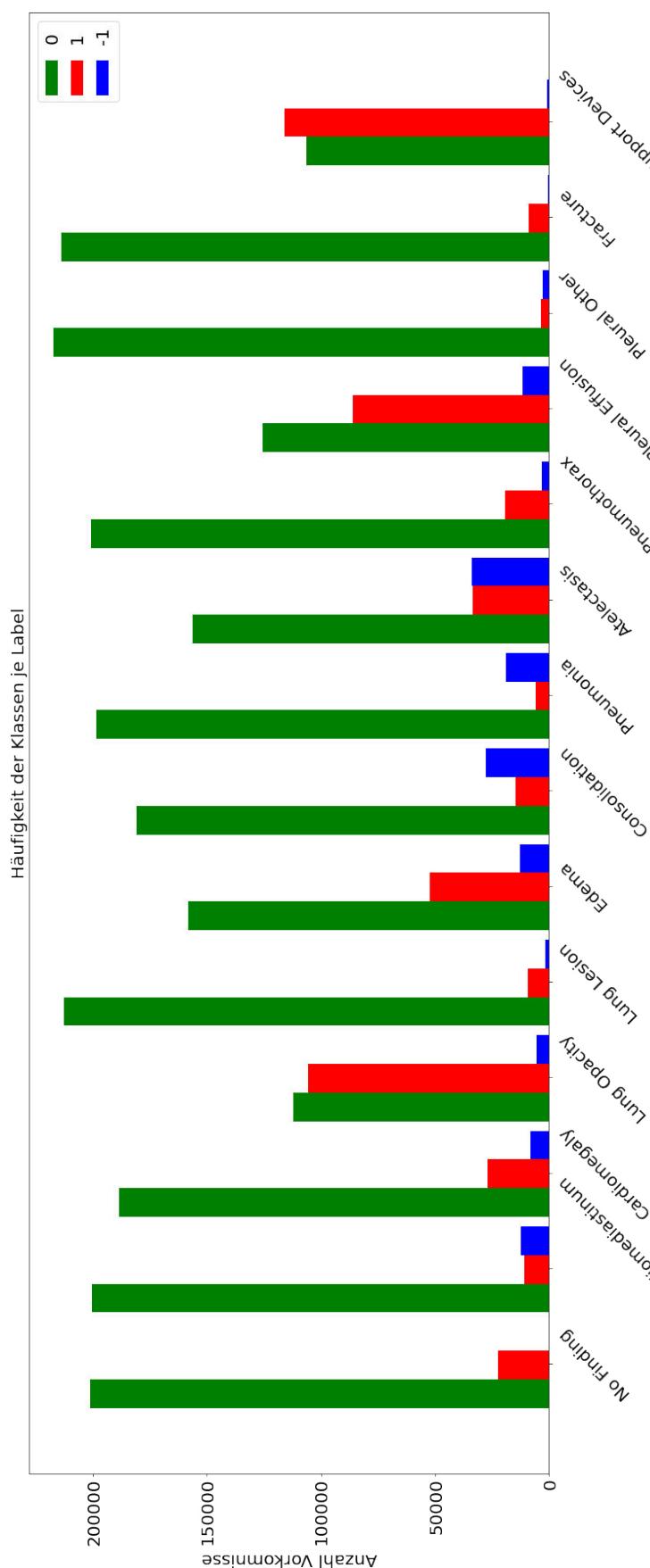
yhat = torch.rand((32, 14))
ytest = (torch.rand(size=(32,14)) < 0.5).int()

%%timeit
customLSEP(yhat, ytest)
13.7 ms ± 250 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

%%timeit
lseploss(yhat, ytest)
96.5 µs ± 2.32 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

```

**Abbildung B.8:** Vergleich der Laufzeit der naiven und Broadcasting PWE-Loss-Implementierung.



**Abbildung B.9:** Häufigkeit der positiven (1, rot), negativen (0, grün) und uncertain (-1, blau) Klasse je Label des CheXpert Trainingsdatensatzes.



**Abbildung B.10:** Chord-Diagramm nach einer Iteration des MLSMOTE-Vorgehens.

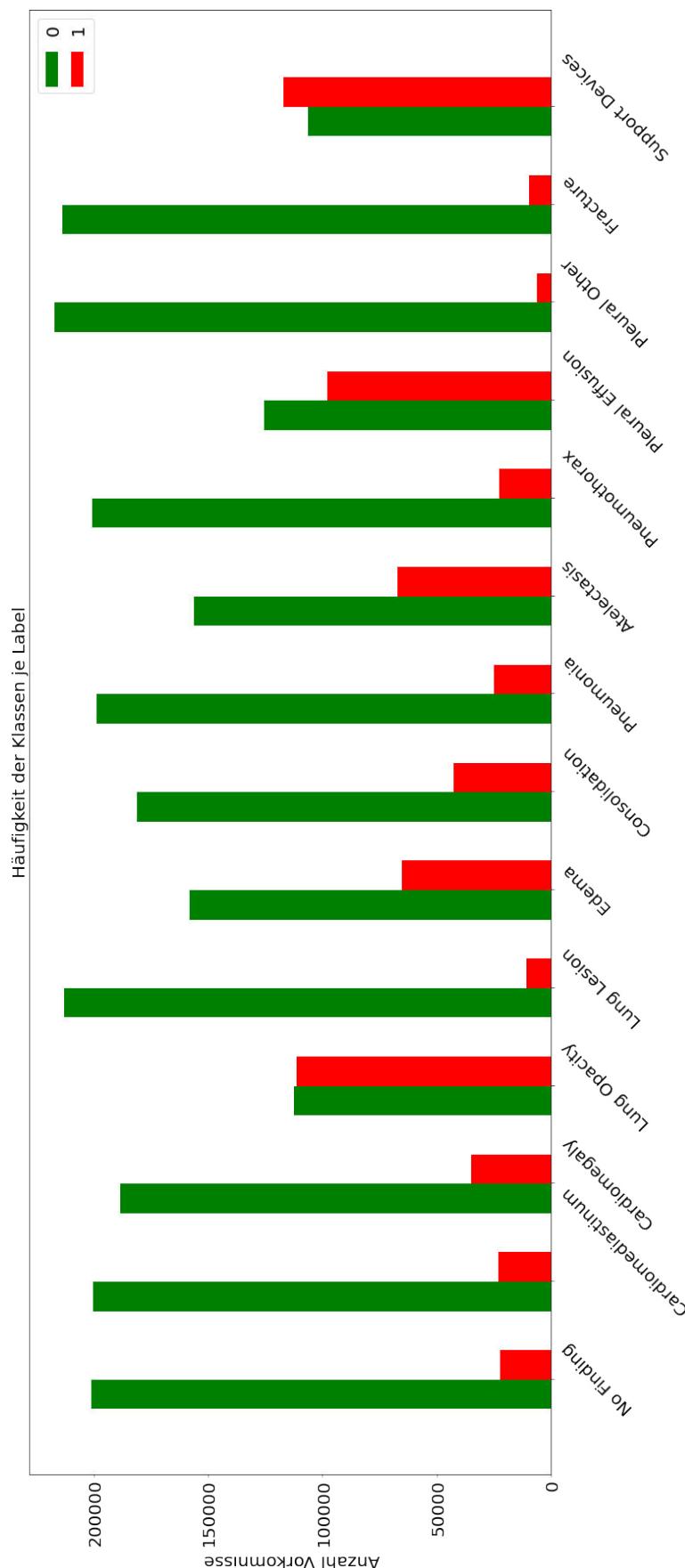
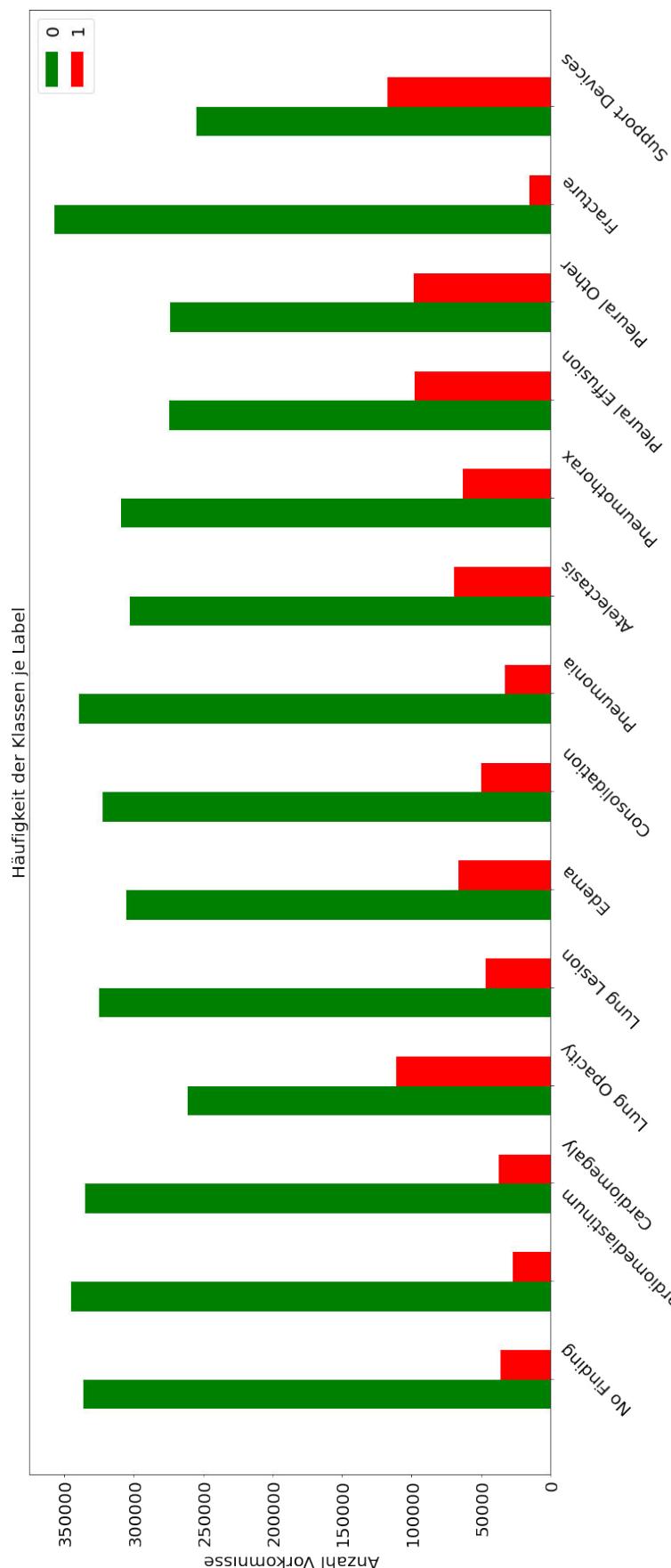
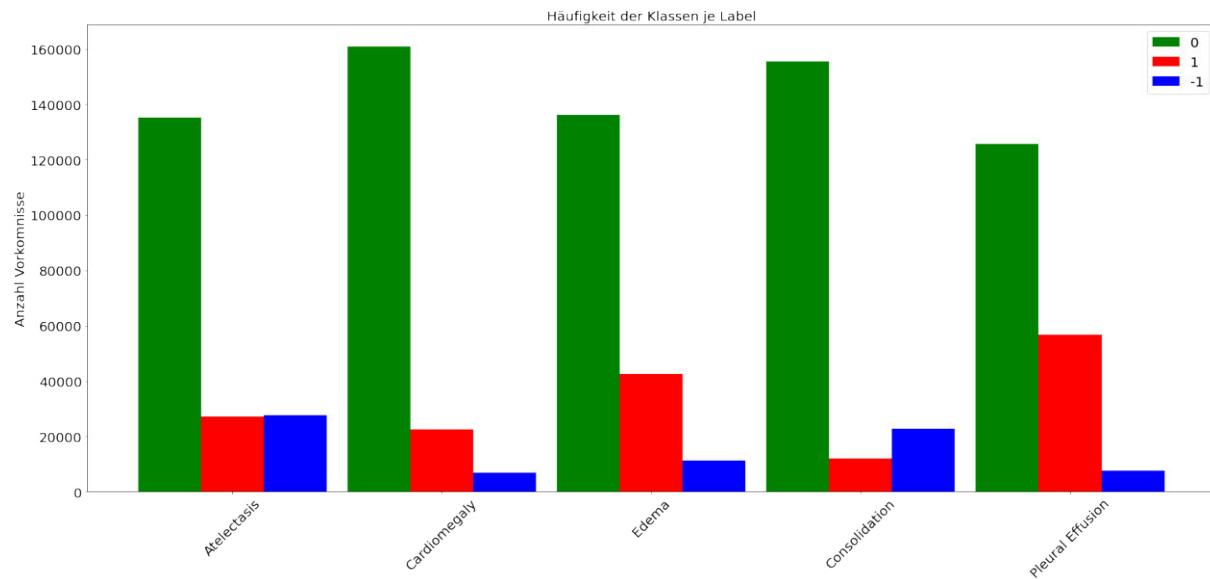


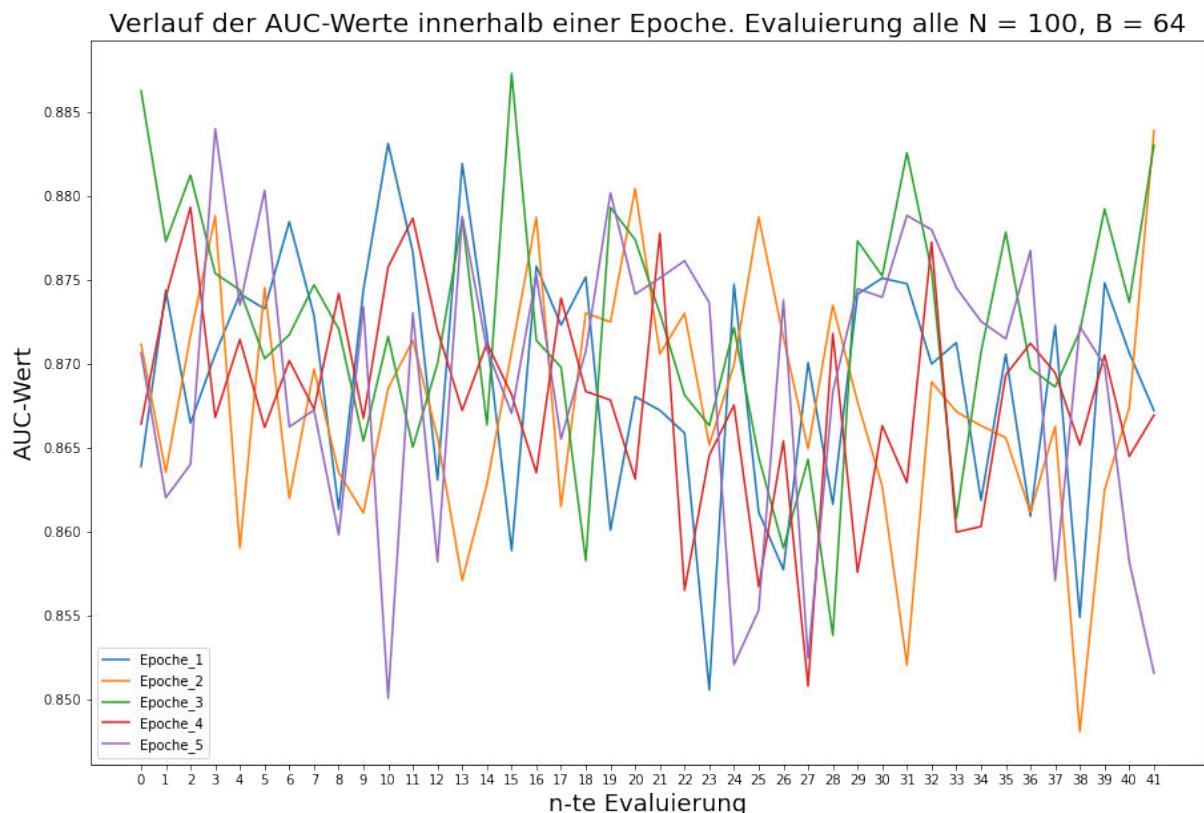
Abbildung B.11: Verteilung der Klassen je Label bei U-Ones Vorgehen.



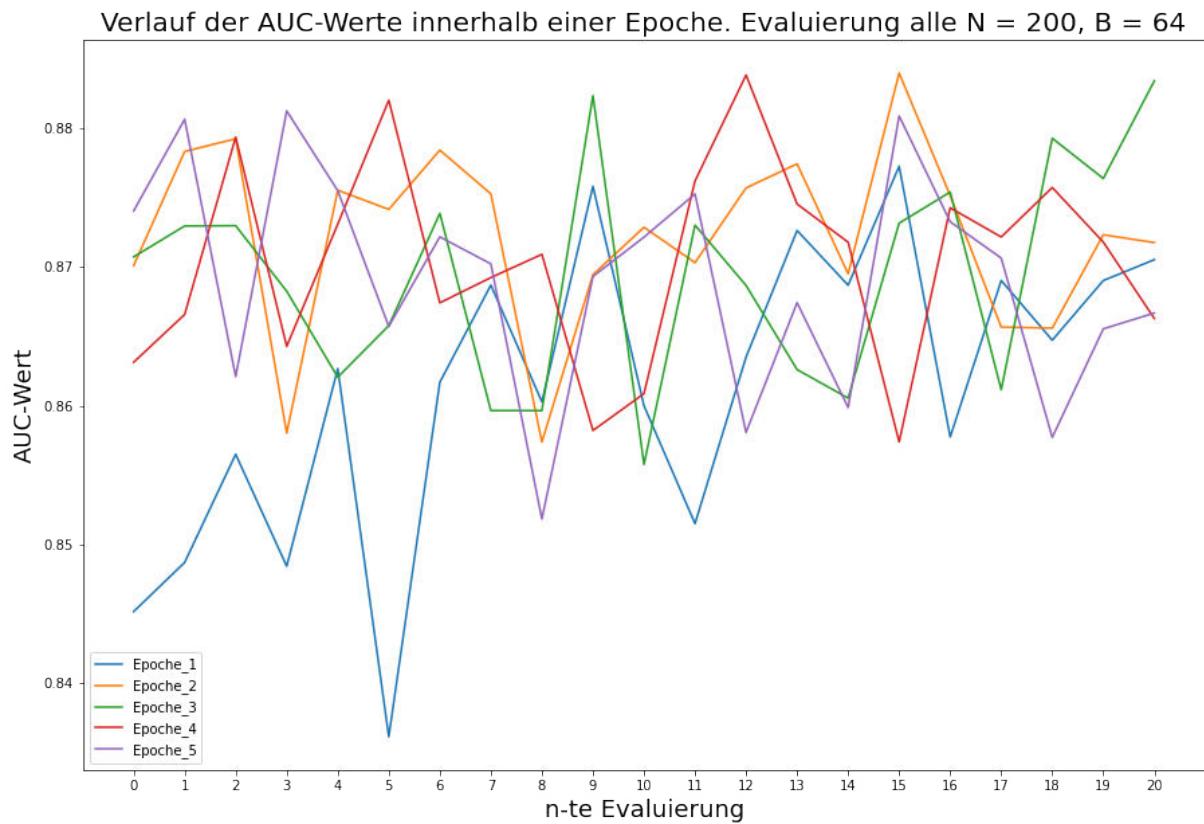
**Abbildung B.12:** Verteilung der Klassen je Label bei U-Ones Vorgehen nach dem Oversampling zum Ausgleich der Imbalance.



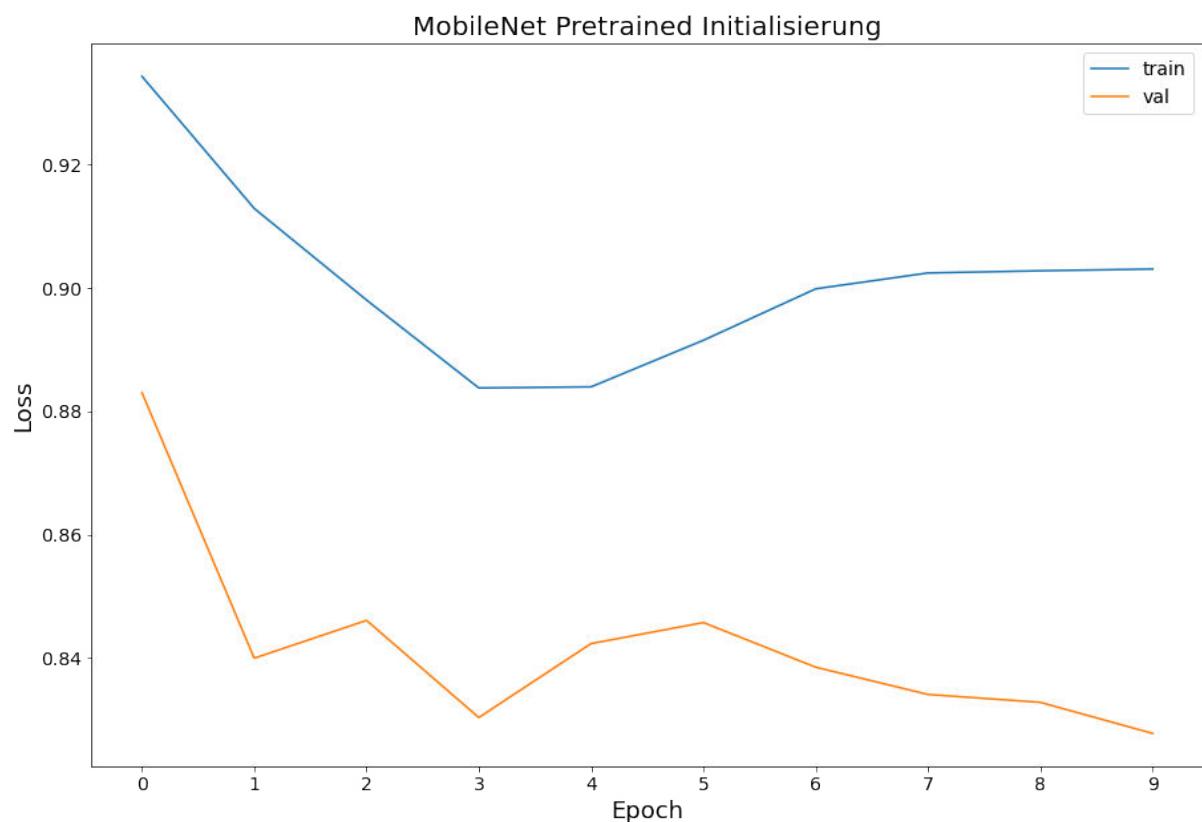
**Abbildung B.13:** Verteilung der Klassen je Label für die fünf Vergleichslabel.



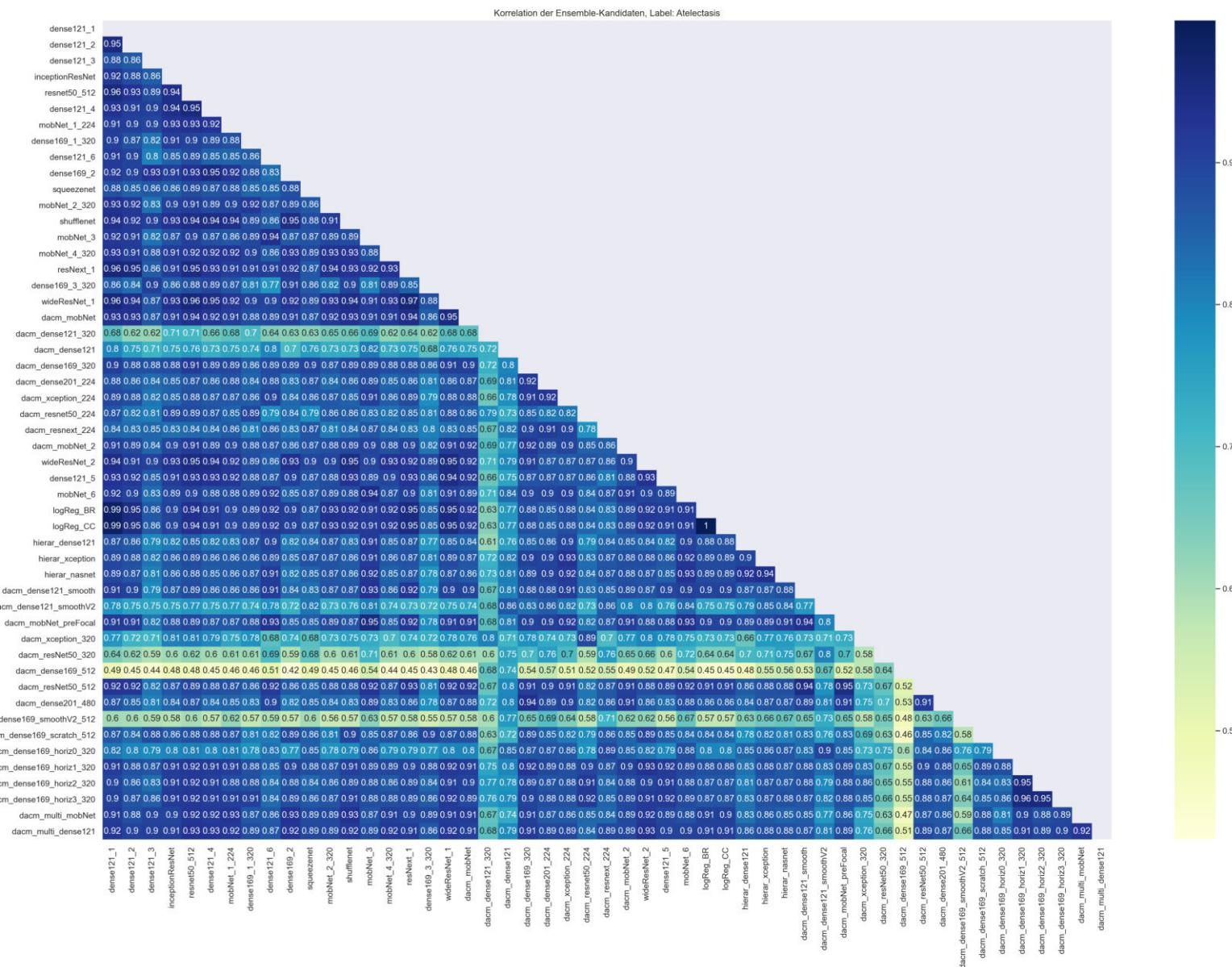
**Abbildung B.14:** Beispiel für die Intra-Epoch Evaluation. AUC-Wert, den ein MobileNet V2 auf den CheXpert Validierungsdaten in der jeweiligen Epoche erreicht. Evaluiert wurde alle 100 Batches bei einer Batchsize von 64.



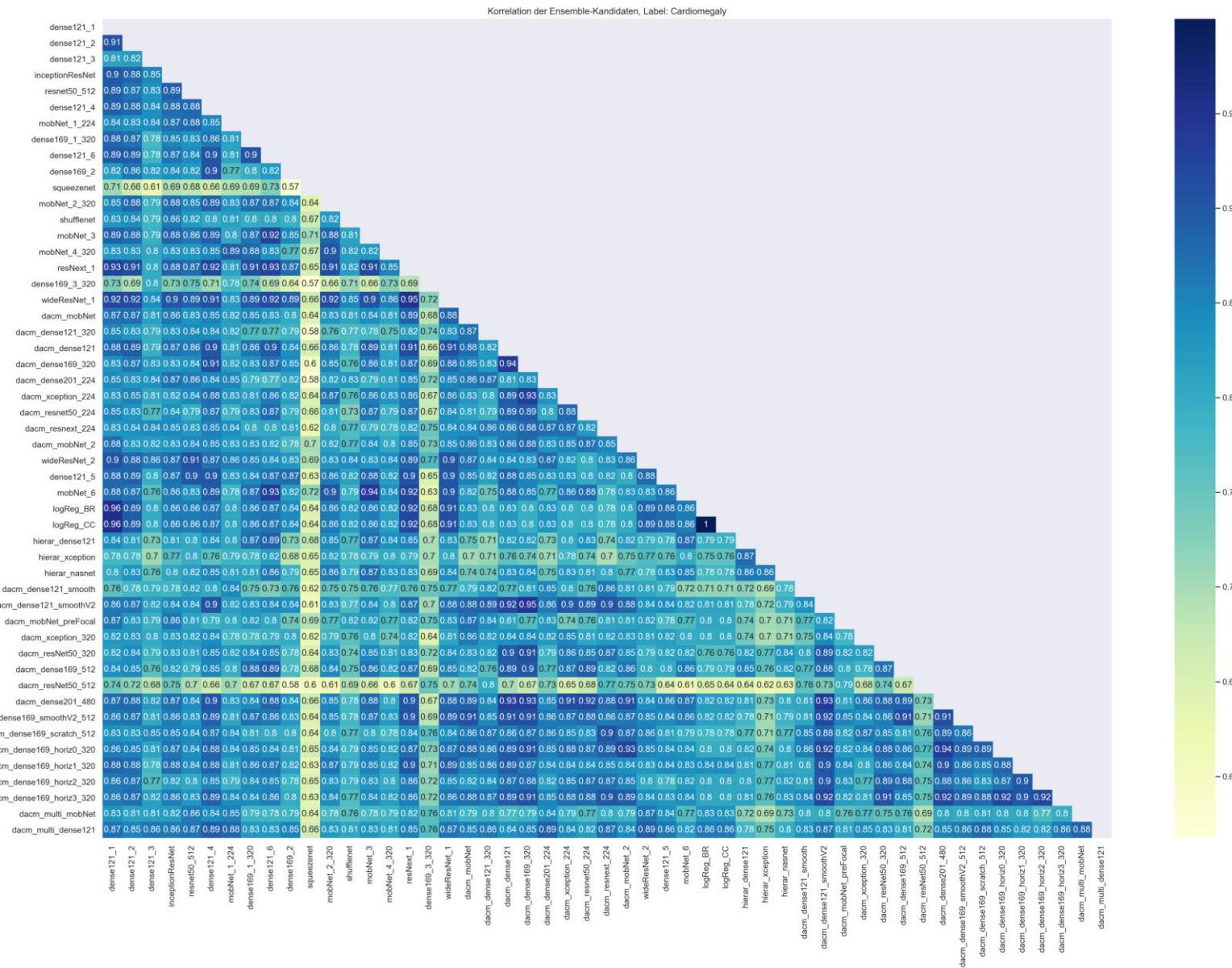
**Abbildung B.15:** Beispiel für die Intra-Epoch Evaluation. AUC-Wert, den ein MobileNet V2 auf den CheXpert Validierungsdaten in der jeweiligen Epoche erreicht. Evaluiert wurde alle 200 Batches bei einer Batchsize von 64.



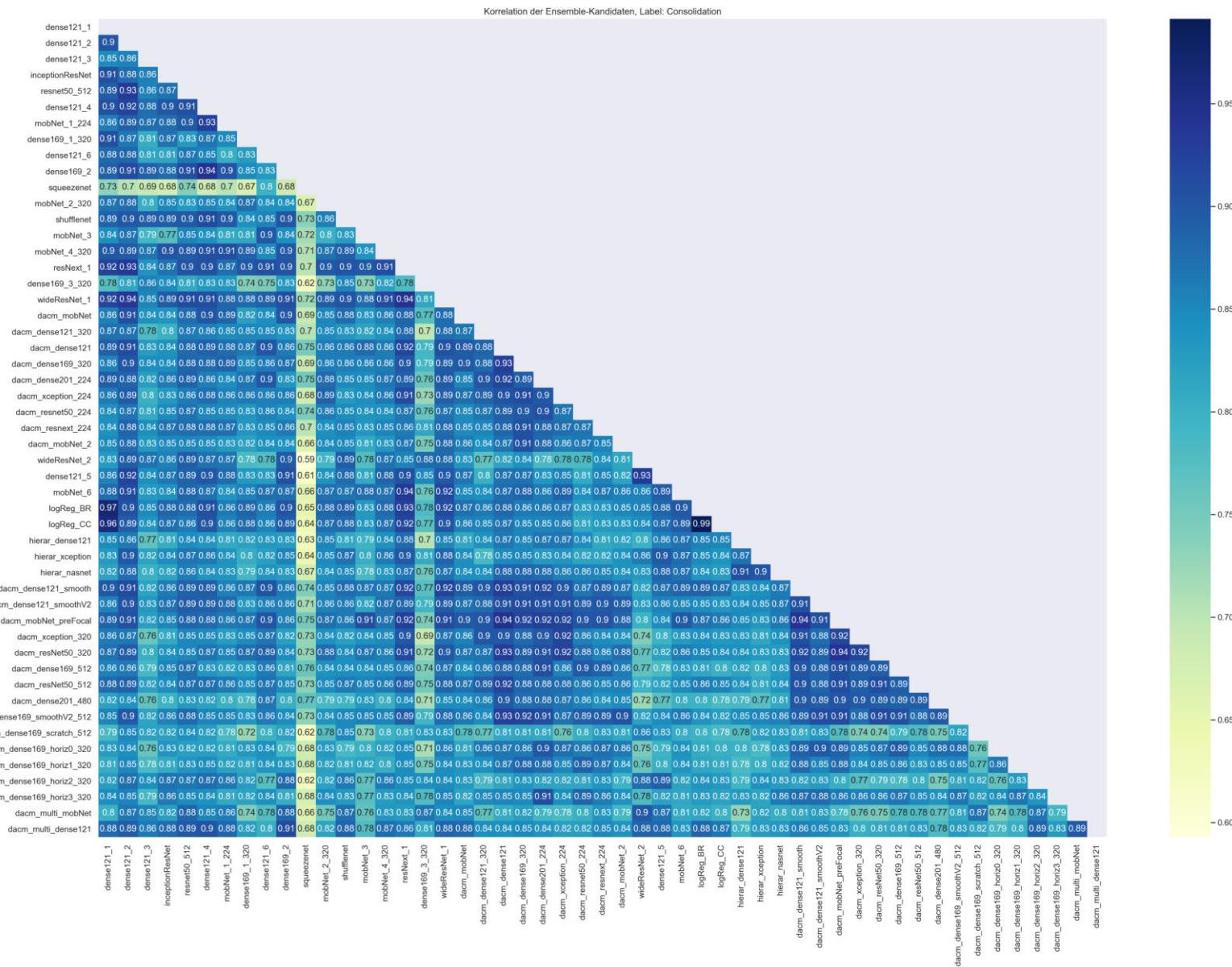
**Abbildung B.16:** Verlauf der Loss-Werte für ein MobileNet V2 mit auf dem ImageNet Datensatz vortrainierten Gewichten



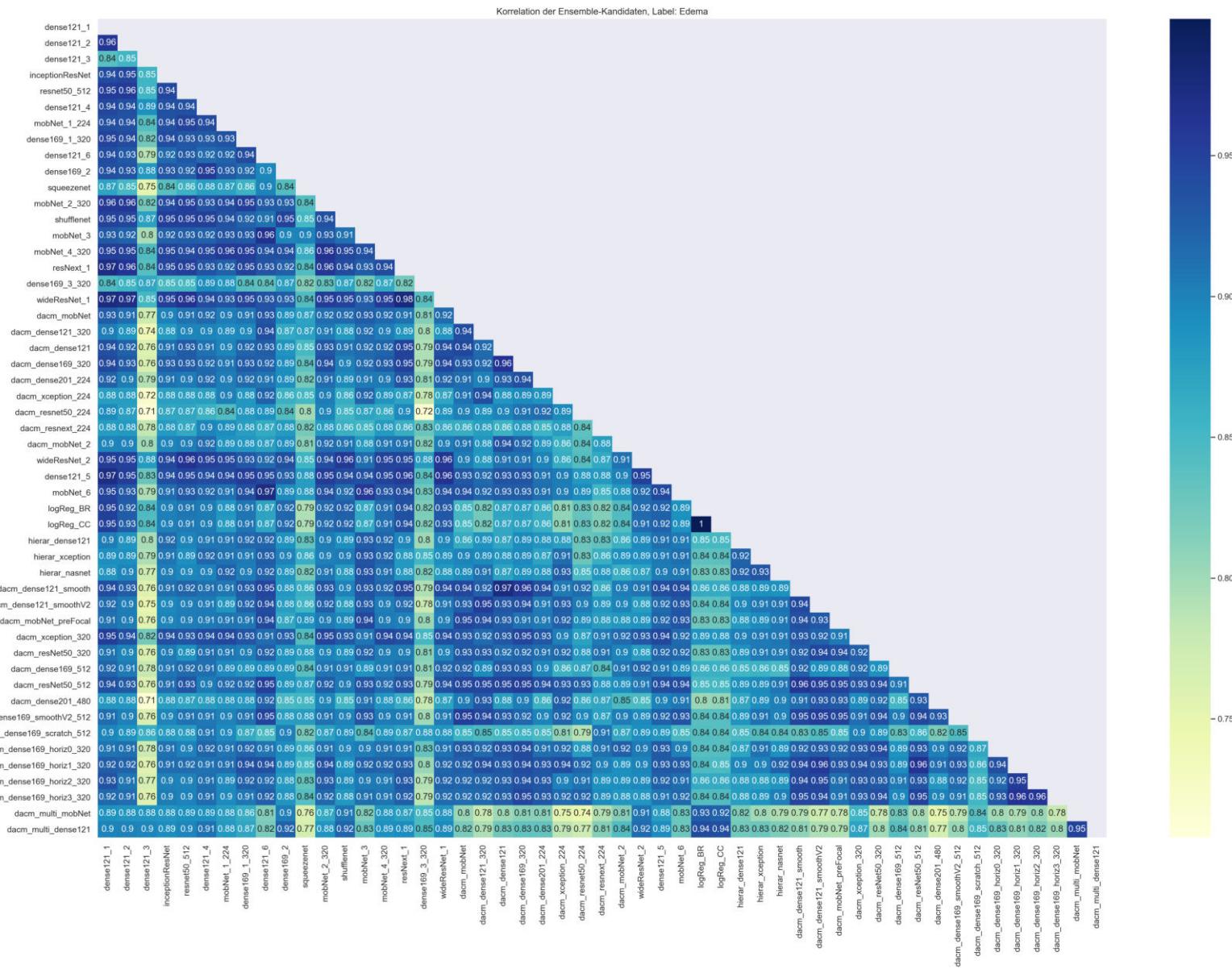
**Abbildung B.17:** Korrelation der Ensemble-Kandidaten für das Label Atelectasis



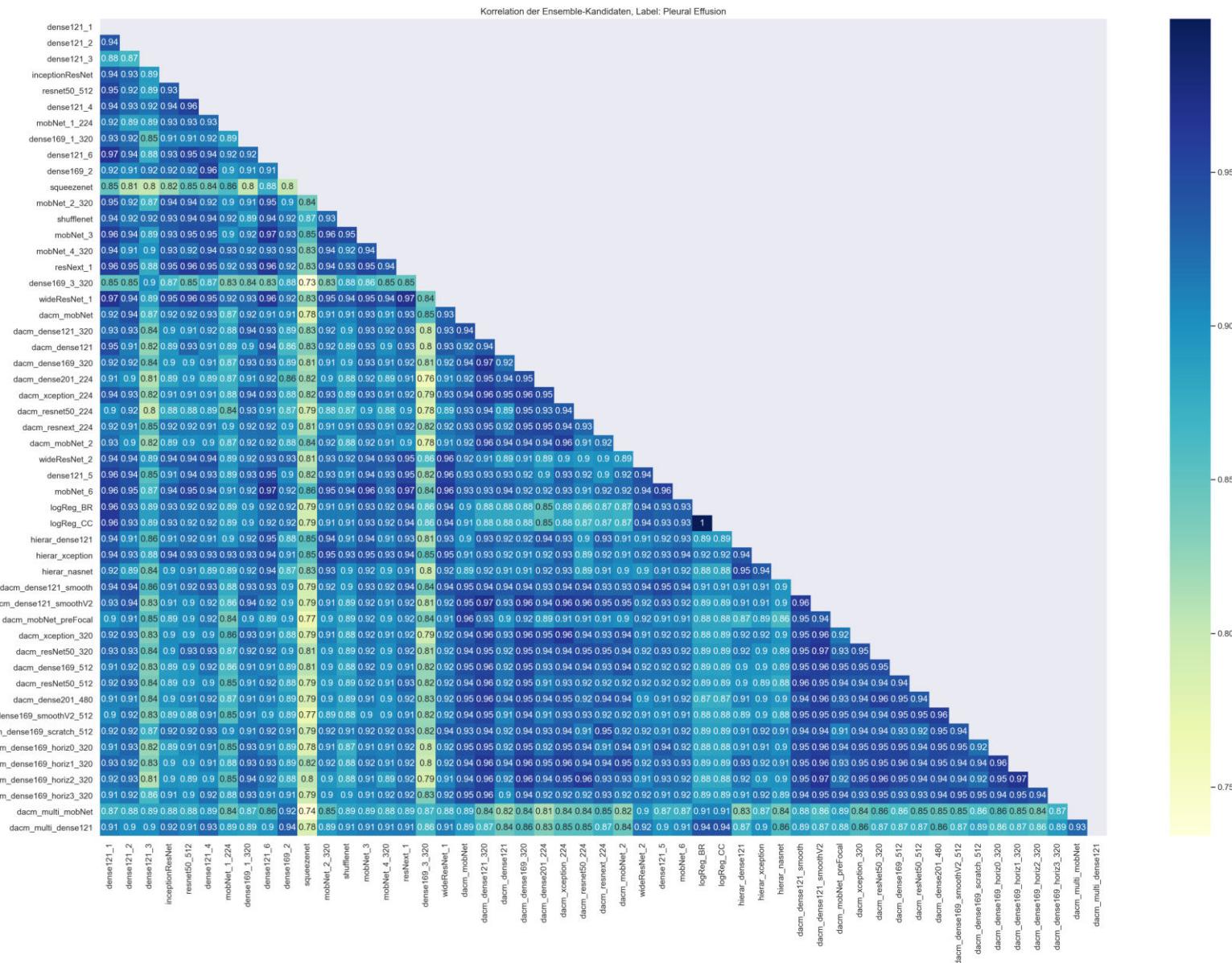
**Abbildung B.18:** Korrelation der Ensemble-Kandidaten für das Label Cardiomegaly



**Abbildung B.19:** Korrelation der Ensemble-Kandidaten für das Label Consolidation



**Abbildung B.20:** Korrelation der Ensemble-Kandidaten für das Label Edema



**Abbildung B.21:** Korrelation der Ensemble-Kandidaten für das Label Pleural Effusion

# Anhang C

## Listings

```

1 def get_activation(dictVar, nameForDict, model):
2 """
3 Adds a forward-hook to the model, which saves the activations under the
4     key nameForDict in the dictionary dictVar
5 """
6     def hook(model, input, output):
7         with torch.no_grad():
8             dictVar[nameForDict] = (torch.cat((dictVar[nameForDict],
9                 output.detach().type(torch.float16).cpu()), dim=0))
10            return hook
11
12 def registerAllHooks(resnet, obj):
13     # Register hook for last Conv Layer in third layer since it's most
14     # effective
15     list(resnet.layer2.children())[-1].conv3.register_forward_hook(get_
16     activation(obj, "L3", resnet))
17
18 # Map to save the activations
19 logits_train = {"L3": torch.Tensor()}
20
21 registerAllHooks(resnet, logits_train)

```

**Listing C.1:** Beispielcode zum Speichern der Aktivierungen

```

1 class FRODO():
2     def __init__(self, training_means):
3         self.scaler = StandardScaler(copy=True, with_mean=True, with_std
4 =True).fit(training_means)
5         self.scaledTrainingFeatures = self.scaler.transform(training_
means)
6         self.clf = ShrunkCovariance(assume_centered=True).fit(self.
scaledTrainingFeatures)
7         self.training_distances = self.clf.mahalanobis(self.
scaledTrainingFeatures)
8         self.trainDistances_mean = self.training_distances.mean()
9         self.trainDistances_std = self.training_distances.std()
10
11     def check(self, logits, n_forStd=2):

```

```

11     means = torch.mean(logits, (2,3))
12     scaled = self.scaler.transform(means)
13     dist = self.clf.mahalanobis(scaled)
14
15     # say its OoD if its more than n_forStd std away from the
16     # training_mean
17     isOoD = np.abs(dist - self.trainDistances_mean) > n_forStd *
18     self.trainDistances_mean
19
20     return isOoD

```

Listing C.2: FRODO-Klasse zur OoD-Detektion

```

1 def forward(self, x: Tensor, returnActivations=False, returnPooling=
2 False) -> Tensor:
3     """
4     x: Input,
5     returnActivations: to return last conv output
6     returnPooling: to return last pool output
7     """
8
9     features = self.features(x)
10    out = F.relu(features, inplace=True)
11    out_pool = F.adaptive_avg_pool2d(out, (1, 1))
12    out = torch.flatten(out_pool, 1)
13    out = self.classifier(out)
14
15    if returnActivations and returnPooling:
16        return out, features, out_pool
17    if returnActivations: #for local branch
18        return out, features
19    if returnPooling: # for fusion branch
20        return out, out_pool
21    return out # general / for global branch

```

Listing C.3: Abgewandelte forward()-Methode der DenseNet-Klasse aus dem torchvision-Paket.

```

1 def getHeatmaps(activations):
2     #Paper: We first take the absolute value of the activation then the
3     #attention heat map Hg is generated by
4     # counting the maximum values along channels
5     return torch.max(torch.abs(activations), dim=1).values
6
7 def resizeHeatmapsBatch(heatmaps, newsize=(224,224)):
8     """
9     Resizes a batch of heatmaps with shape Batch x H x W without
10    torchvision/transforming to PIL
11
12    hms = heatmaps.shape
13    if (hms[1] == 0) or (hms[2] == 0):
14        return None
15    return F.interpolate(
16        heatmaps.reshape(hms[0], 1, hms[1], hms[2]), # reshape to B x 1
17        x H x W
18        size=newsize, mode="bilinear", # torchvision uses bilinear mode
19        align_corners=False # suppress warning

```

```
17 ).reshape(hms[0], *newsize) # reshape to correct dimensions
```

**Listing C.4:** Funktionen zur Erzeugung und Skalierung der Heatmaps.

```
1 def getLargestConnectedAreaMask(singleResizedHeatmap, thresh=None,
2                                 useThresholdOtsu=False):
3     """
4     Gets the largest connected area as a mask for a single resized
5     heatmap
6     """
7     image = singleResizedHeatmap.cpu().numpy()
8     if thresh is None:
9         if useThresholdOtsu:
10            thresh = threshold_otsu(image)
11        else:
12            thresh = np.mean(image) + np.std(image)
13    # apply threshold and fill small holes
14    bw = closing(image > thresh, square(3))
15    # remove artifacts connected to image border
16    cleared = clear_border(bw)
17    # label image regions
18    label_image, num = skimage_label(cleared, return_num=True)
19    # num is number of regions but also the index of the largest region
20    mask = (label_image == num)
21    return torch.Tensor(mask)
22
23 def normToZeroOne(batch):
24     """
25     Normalizes the batch to have values between [0, 1]
26     """
27     view = batch.view(batch.shape[0], -1)
28     bmin = view.min(1, keepdim=True).values
29     bmax = view.max(1, keepdim=True).values
30
31     return (batch - bmin.unsqueeze(1)) / (bmax - bmin).unsqueeze(1)
32
33 def getMasksFromHeatmaps(resizedHeatmaps, thresh=0.7):
34     """
35     Normalizes the heatmaps to [0, 1] range and then iteratively gets
36     the masks. Default
37     threshold = 0.7 like in the paper
38     """
39     masks = []
40     normed = normToZeroOne(resizedHeatmaps)
41     for hm in normed:
42         m = getLargestConnectedAreaMask(hm, thresh=thresh)
43         masks.append(m.unsqueeze(0).unsqueeze(0)) # batches x channel x
44         h x w
45     return torch.cat(masks) # concat along batch dim
```

**Listing C.5:** Erzeugen der Masken aus den Heatmaps.

```
1 def cutoutSquareByMask(img, mask):
2     """
3     For a Batch: Cuts out the maximum connected area (square) covered by
4     the mask from the image
```

```

4   Inputs:
5     - img: Image that should be cutout
6     - mask: binary ROI mask, 1:keep, 0: discard
7
8   Outputs:
9     - the squared mask cutout of the image
10    """
11
12    batch_size, channels = img.shape[0], img.shape[1]
13    allCutouts = torch.zeros((batch_size, channels, img.shape[2], img.
14      shape[3])) # storage for cutouts
15
16    for i in range(batch_size):
17      m = mask[i]
18      c, h, w = torch.where(m == 1)
19      w_min, w_max = torch.min(w), torch.max(w)
20      h_min, h_max = torch.min(h), torch.max(h)
21      single_cutout = img[i][:, h_min:h_max, w_min:w_max]
22      resized_cutout = resizeHeatmapsBatch(single_cutout)
23      if resized_cutout is None:
24        allCutouts[i] = img[i]
25      else:
26        allCutouts[i] = resized_cutout
27
28  return allCutouts

```

**Listing C.6:** Ausschneiden der quadratischen ROI aus dem Bild.

```

1 class Identity(nn.Module):
2   def __init__(self):
3     super(Identity, self).__init__()
4
5   def forward(self, x):
6     return x

```

**Listing C.7:** Identitäts-Schicht, die die Eingabe unverändert zurückgibt.

```

1 class CascadeSingleLevel(nn.Module):
2   def __init__(self, level, n_classes = 14, usePretrained=False):
3     super().__init__()
4     self.lvl = level
5     self.resnet = tv.models.resnet50(pretrained=usePretrained)
6     self.resnet.fc = nn.Linear(in_features= self.resnet.fc.in_features,
7       out_features=n_classes)
8     if self.lvl > 1:
9       self.dropout = nn.Dropout(p=0.2)
10      self.fcOut = nn.Linear(in_features=n_classes+((self.lvl-1)*n_
11        classes), out_features=n_classes)
12
13  def forward(self, X, prevOutput):
14    out = self.resnet(X)
15    if self.lvl <= 1:
16      return out
17    out = self.dropout(out)
18    out = torch.cat([out, *prevOutput], dim=1) # dim 0 is batch, unpack
19    prevOutput
20    out = out.view(-1, self.fcOut.in_features)

```

```

18     out = self.fcOut(out)
19
20     return out

```

**Listing C.8:** Klasse zur Erzeugung eines Kaskaden-Objektes eines beliebigen Levels.

```

1 def customLSEP(output, target, takeAverage=True):
2     batch_size = output.shape[0]
3     running_loss = 0
4     for batch_index in range(batch_size):
5         u = (target[batch_index] == 1.)
6         v = (target[batch_index] == 0.)
7         u_yhat = yhat[batch_index][u]
8         v_yhat = yhat[batch_index][v]
9         batchSum = 0
10        for vi in v_yhat:
11            for ui in u_yhat:
12                batchSum+=torch.exp(vi - ui)
13
14        running_loss += torch.log(1+batchSum)
15    if takeAverage:
16        return running_loss / batch_size
17    else:
18        return running_loss

```

**Listing C.9:** Naive Implementierung des PWE Loss.

```

1 def lsep_loss(input, target, average=True):
2     # https://github.com/ex4sperans/freesound-classification
3     # calculate the differences of all inputs with themselves making use
4     # of broadcasting at newly added
5     # dim 1 and 2, new dim: batch_size x n_classes x n_classes
6     differences = input.unsqueeze(1) - input.unsqueeze(2)
7
8     # again, use broadcasting now to find where pairwise labels are
9     # different. then binarize the mask with float()
10    where_different = (target.unsqueeze(1) < target.unsqueeze(2)).float()
11
12    # only calculate the exp of masked differences, thus only where labels
13    # are pairwise different
14    exps = differences.exp() * where_different
15    # summing + rest of formula
16    lsep = torch.log(1 + exps.sum(2).sum(1))
17    # mean corresponds to devision by batch_size
18    if average:
19        return lsep.mean()
20    else:
21        return lsep

```

**Listing C.10:** Broadcasting Implementierung des PWE Loss aus [113].

## Anhang D

### Inhaltsverzeichnis des Datenträgers

**Masterarbeit.pdf** Diese Arbeit als PDF-Datei.

**DatensatzCSVs** CSVs der modifizierten Datensätze.

**Code** IPython-Notebooks und Python-Dateien.

**Code/AGUtils.py** Hilfsfunktionen für das AG-CNN.

**Code/DeepAUCUtils.py** Hilfsfunktionen für den DAM-Loss.

**Code/utils.py** Allgemeine Hilfsfunktionen.

**Code/\*Baseline-\*.ipynb** Training verschiedener Architekturen, Konfigurationen und Besonderheiten.

**Code/ChordDiagram.ipynb** Notebook zum Erstellen der Chord-Diagramme.

**Code/Clean-FRODO.ipynb** Notebook für die FRODO-Methode.

**Code/Clean-LabelFocused\*.ipynb** Notebooks für die Methode aus [95].

**Code/Clean-LogRegExtractor.ipynb** Untersuchung der Methode aus [52] inkl. PCA.

**Code/Clean-NEW-HorizontalVoting\*.ipynb** Experimente zum Snapshotting.

**Code/EnsembleSearch.ipynb** Festlegen des finalen Ensembles und Untersuchung der Korrelationen.

**Code/EvaluateOnNIh\*ipynb** Evaluieren des finalen Ensembles auf dem ChestX-ray14Datensatz.

**Code/Experiment-IntraEpochEval.ipynb** Experiment zur Intra-Epoch Evaluation.

**Code/ExtendedDatasetMLSMOTE.ipynb** Experiment zu ML-SMOTE.

**Code/FineTune-\*-WithLibAUC\*.ipynb** Finetunen von zuvor trainierten Modellen mittels DAM-Loss / LibAUC-Bibliothek.

**Code/imageGenerationNotebook.ipynb** Untersuchen der Datensatz-Imbalance, Bild-  
derzeugung.

**Code/LibAUCExample\*.ipynb** Training verschiedener Architekturen mittels DAM-Loss / LibAUC-Bibliothek.

**Code/New\_Multilabel\_LibAUC\*.ipynb** Training verschiedener Architekturen mittels Multilabel DAM-Loss / LibAUC-Bibliothek.

**Code/submitCheX.py** Skript für die Einreichung bei der CheXpert Competition.

**Code/TemplateMatching.ipynb** Notebook zum Template Matching.

**Modelle** Die trainierten Gewichte des finalen Ensembles.

**Latex** Latex-Quelle.

**Sonstiges** Chord-Diagramme, Klassenhäufigkeit je Label, Templates, weiterer Code.