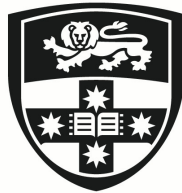


Deep Learning in Chest Radiography: From Report Labeling to Image Classification

MARAM MAHMOUD A. MONSHI

Doctor of Philosophy



THE UNIVERSITY OF
SYDNEY

Research Supervisor: Dr. Josiah Poon
Associate Supervisor: Dr. Vera Chung

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

November 2022

Abstract

Chest X-ray (CXR) is the most common examination performed by a radiologist. Through CXR, radiologists must correctly and immediately diagnose a patient's thorax to avoid the progression of life-threatening diseases. Not only are certified radiologists hard to find but also stress, fatigue, and lack of experience all contribute to the quality of an examination. As a result, providing a technique to aid radiologists in reading CXRs and a tool to help bridge the gap for communities without adequate access to radiological services would yield a huge advantage for patients and patient care. This thesis considers one essential task, CXR image classification, with Deep Learning (DL) technologies from the following three aspects: understanding the intersection of CXR interpretation and DL; extracting multiple image labels from radiology reports to facilitate the training of DL classifiers; and developing CXR classifiers using DL.

First, we explain the core concepts and categorize the existing data and literature for researchers entering this field for ease of reference. Using CXRs and DL for medical image diagnosis is a relatively recent field of study because large, publicly available CXR datasets have not been around for very long. The current understanding of radiology text and image structures, application of DL algorithms, utilization of available datasets, labeling of reports, classification of images, generation of reports, and evaluation of models are thoroughly investigated to evaluate the strengths and limitations of what has been done in relation to this thesis.

Second, we contribute to the understanding of one of the primary challenges in the development of CXR classification models, which is labeling large datasets with multi-label image annotations extracted from radiology reports. We describe the development of a DL-based report labeler model named CXRlabeler, focusing on inductive sequential transfer learning. To the best of our knowledge, CXRlabeler is the first proposed model that has the benefits of both Language Model (LM) fine-tuning and classifier fine-tuning to achieve

highly accurate automated CXR report labeling. It fine-tunes a pre-trained LM to the corpus of radiology impressions and then uses the LM encoder with a new head to simultaneously extract observations from free-text CXR reports.

Lastly, we explain the design of three novel Convolutional Neural Network (CNN) classifiers, i.e., MultiViewModel, Xclassifier, and CovidXrayNet, for binary image classification, multi-label image classification, and multi-class image classification, respectively. MultiViewModel introduces a stage-wise training technique and combines frontal and lateral CXRs to improve the performance of thoracic disease detection. Xclassifier is proposed based on distributed DL methods and an anti-aliasing filter to reduce the computational complexity while preserving the classifier accuracy. For the multi-class image classification problem, we propose CovidXrayNet to classify a CXR into one of the following classes: coronavirus disease 2019, normal, or pneumonia. It focuses on optimizing CNN hyperparameters and data augmentations and achieves high accuracy on the benchmark dataset and our generated balanced COVIDcxr dataset, with only a few epochs of training.

This dissertation showcases significant progress in the field of automated CXR interpretation using DL; all source code used is publicly available. It provides methods and insights that can be applied to other medical image interpretation tasks.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work.

This thesis has not been submitted for any degree or other purpose.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis has been acknowledged.

Student: Maram Mahmoud A. Monshi

Signature:

Date: November 9 , 2022

Acknowledgements

This thesis is dedicated to my parents, the chemistry professor Mahmoud Monshi (1949–2018) and the retired primary school teacher Samiera Sendy, who instilled a passion for learning in me at an early age. I want to thank my mother for all the love, care, encouragement, and support she has given me over the years despite the geographical distance between us and the loss of her husband. My father passed away in the fifth month of my PhD journey; may Allah bless his soul and grant him paradise.

I would like to express my immense gratitude to my research supervisor, Dr. Josiah Poon, for his supervision, encouragement, support, and constructive guidance throughout my PhD research. His constant advice and feedback pushed me to sharpen my thinking and bring my research to a higher level. I am incredibly grateful for our regular meetings and conversations, which were vital in inspiring me to think outside the box, and from multiple perspectives to conduct comprehensive and objective research.

My grateful thanks go to my auxiliary supervisor, Dr. Vera Chung, for her constructive comments throughout each stage of the process, all of which contributed significantly to the final product of this thesis.

I would like to acknowledge Taif University in Saudi Arabia for providing me with scholarship funding and thank Saudi Arabian Culture Mission (SACM) in Australia for their help and support throughout the years. I thank Google Cloud research credits program for supporting me to advance my research by giving me access to the computing power that made this thesis possible. I also thank the school of computer science at the University of Sydney for providing me with ongoing study and research resources, access to advanced facilities, and a great study space, where most of this thesis was written. I also want to thank them for the yearly progress evaluation, which helped me get constructive feedback from my supervisory team and review panel and finish my research on time. Thanks to Proofed Inc. for proofreading this thesis.

From the bottom of my heart, sincere and special thanks go to my husband and best friend, Ibraheem, for encouraging me daily to pursue all my dreams, including my bachelor's degree and master's degree and this PhD dissertation.

A special thanks to my heart's angels, Albaraa, AbdulRahman, Yousef, Byan, and Basma, for all of their innocent love and support. I thank them for their creative letters, soft kisses, and warm hugs, which made me happier, more determined, and stronger. I thank them for their patience in waiting for quality time with their mother. I thank my two oldest boys, Albaraa and AbdulRahman, for being responsible, understanding my preoccupation, and cooperating by studying hard during their high school years. I especially thank Yousef and Byan for providing happy distractions to rest my mind outside of my research. I thank Basma, a name of Arabic origin that means "smile," for being the most beautiful part of my PhD degree, as I welcomed her in the second month of my PhD journey. Her sweet smile helped ease my stress over the years, and her ability to understand three languages by the age of three was a great inspiration and offered good insight into how the human brain can learn and think.

My thanks go to my siblings, Adil, Emad, Manal, Manahil, and Fahad, who became doctorate holders in their fields before me, for supporting me emotionally to overcome the loss of our beloved dad and for being there every time I needed one of them throughout my PhD journey. Special thanks to my youngest brother, the radiologist Fahad, for the many inspiring Zoom calls in which he helped me bridge the gap between computer science and radiology.

Finally, I would like to express my most profound appreciation to my second family in Sydney: Sania Azhar's family. This work could not have been completed without their extraordinary support and love.

List of Publications

Journals

- J1 Monshi, M.**, Poon, J., Chung, V. (2020). Deep learning in generating radiology reports: A survey. *Artificial Intelligence in Medicine*, 106, 101878. (CORE: A, Impact Factor: 4.383) [\[Paper\]](#) [\[Citation: 74\]](#)
- J2 Monshi, M.**, Poon, J., Chung, V., Monshi, F. (2021). CovidXrayNet: Optimizing data augmentation and CNN hyperparameters for improved COVID-19 detection from CXR. *Computers in Biology and Medicine*, 133, 104375. (CORE B, Impact Factor: 3.434). [\[Paper\]](#) [\[Code\]](#) [\[Citation: 43\]](#)

Conferences

- C1 Monshi, M.**, Poon, J., Chung, V. (2019). Convolutional Neural Network to Detect Thorax Diseases from Multi-view Chest X-Rays. 26th International Conference on Neural Information Processing (ICONIP 2019), Cham: Springer International Publishing. (CORE A). [\[Paper\]](#) [\[Code\]](#) [\[Citation: 8\]](#)
- C2 Monshi, M.**, Poon, J., Chung, V., Monshi, F. (2021), “Labeling Chest X-Ray Reports Using Deep Learning,” in International Conference on Artificial Neural Networks (ICANN), 2021 (CORE C). [\[Paper\]](#) [\[Code\]](#) [\[Citation: 1\]](#)
- C3 Monshi, M.**, Poon, J., Chung, V. (2022). Distributed Deep Learning for Multi-Label Chest Radiography Classification, 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), 2022 (CORE B). Nominated for Best Poster Award. [\[Paper\]](#) [\[Code\]](#) [\[Citation: 1\]](#)

Authorship Attribution Statement

Chapter 2 of this thesis is published as **J1**.

I am the first and corresponding author. I conceived and designed the review, analyzed the data, and wrote the entire manuscript, from the initial draft to the published version.

Chapter 3 of this thesis is published as **C2**.

I am the first and corresponding author. I conceived, designed, and performed the experiments, analyzed the data, wrote the entire manuscript, from the initial draft to the published version, and presented the paper at the conference.

Chapter 4 of this thesis is published as **C1**.

I am the first and corresponding author. I conceived, designed, and performed the experiments, analyzed the data, wrote the entire manuscript, from the initial draft to the published version, and presented the paper at the conference.

Chapter 5 of this thesis is published as **C3**.

I am the first and corresponding author. I conceived, designed, and performed the experiments, analyzed the data, wrote the entire manuscript, from the initial draft to the published version, and presented the paper at the conference.

Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
AG-CNN	Attention-Guided Convolutional Neural Network
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Anteroposterior
AUC	Area Under the Receiver Operating Characteristic Curve
AWD-LSTM	Averaged Stochastic Gradient Descent Weight-Dropped Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CAD	Computer-Aided Detection
CIDEr	Consensus-Based Image Description Evaluation
CNN	Convolutional Neural Network
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
CT	Computed Tomography
CXR	Chest X-ray
Caffe	Convolutional Architecture for Fast Feature Embedding
DDP	Distributed Data Parallel
DDSM	Digital Database for Screening Mammography
DICOM	Digital Imaging and Communications in Medicine
DL	Deep Learning
DP	Data Parallel
DenseNet	Densely Connected Convolutional Network
FLOPS	Floating-Point Operations per Second
GEV	Generalized Extreme Value

GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HRGR-Agent	Hybrid Retrieval-Generation Reinforced Agent
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IU X-Ray	Indiana University Chest X-Ray
JPEG	Joint Photographic Experts Group
KNN	K-Nearest Neighbor
LDA	Latent Dirichlet Allocation
LDPO	Looped Deep Pseudo-Task Optimization
LM	Language Model
LR	Learning Rate
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MIMIC-CXR	Medical Information Mart for Intensive Care Chest X-ray
ML	Machine Learning
MRI	Magnetic Resonance Imaging
MeSH	Medical Subject Headings
NIH	National Institute of Health
NIN	Network in Network
NLM	National Library of Medicine
NLP	Natural Language Processing
PACS	Picture Archiving and Communication System
PA	Posteroanterior
PEIR	Pathology Education Informational Resource
PET	Positron Emission Tomography
PPV	Positive Predictive Value
PadChest	PAthology Detection in Chest radiographs
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RNN-ATT	Recurrent Neural Network with Attention
ROIs	Rectangular Regions of Interest
ROUGE	Recall-Oriented Understudy for Gisting Evaluation

RSNA	Radiological Society of North America
RT-PCR	Reverse Transcription Polymerase Chain Reaction
ReLU	Rectified Linear Unit
ResNet	Residual Network
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SENet	Squeeze and Excitation Network
SGD	Stochastic Gradient Descent
SIRM	Italian Society of Medical, Interventional Radiology
SPICE	Semantic Propositional Image Caption Evaluation
SVM	Support Vector Machine
TanH	Hyperbolic Tangent
TieNet	Text–Image Embedding Network
US	Ultrasound
VGG	Visual Geometry Group

Contents

Abstract	ii
Statement of Originality	iv
Acknowledgements	v
List of Publications	vii
Authorship Attribution Statement	viii
Acronyms	x
Contents	xiii
List of Figures	xviii
List of Tables	xx
Chapter 1 Introduction	1
1.1 Background	1
1.2 Motivation	3
1.3 Contributions	4
1.4 Outline	6
Chapter 2 Literature Review	7
2.1 Introduction	7
2.1.1 Contributions	8
2.2 Radiology	8
2.2.1 Understanding radiology text	10
2.2.2 Understanding radiology images	11
2.2.3 Understanding CXR findings	13

2.3	Deep learning	16
2.3.1	Activation function	18
2.3.2	Convolutional neural network	18
2.3.2.1	Architecture	20
2.3.3	Recurrent neural networks	22
2.3.4	Data augmentation	23
2.3.5	Hyperparameters	23
2.3.6	Software	24
2.4	CXR datasets	25
2.4.1	IU X-Ray	25
2.4.2	ChestX-ray14	27
2.4.3	CheXpert	27
2.4.4	MIMIC-CXR	28
2.4.5	PadChest	30
2.4.6	COVID-19 datasets	30
2.4.7	Private datasets	32
2.4.8	Beyond CXR	32
2.5	CXR report labeling	32
2.5.1	Feature engineering approaches	33
2.5.2	Deep learning approaches	34
2.5.3	Beyond report labeling	35
2.6	CXR image classification	36
2.6.1	Binary	36
2.6.2	Multi-label	38
2.6.3	Multi-class	38
2.6.4	Beyond CXR classification	39
2.7	CXR computer-aided applications	44
2.8	Evaluation	45
2.8.1	Quantitative classification metrics	46
2.8.2	Quantitative captioning metrics	48

2.8.3	Qualitative measures	51
2.9	Discussion and future directions	51
2.10	Conclusion	53
Chapter 3	Report Labeling	54
3.1	Introduction	54
3.1.1	Contributions	56
3.2	Proposed CXRlabeler model	56
3.2.1	Data preparation	58
3.2.2	Language model	59
3.2.3	Multi-label classifier	60
3.3	Experiment	60
3.4	Results and discussion	61
3.5	Summary and conclusion	64
Chapter 4	Binary Image Classification	65
4.1	Introduction	65
4.1.1	Contributions	66
4.2	Proposed MultiViewModel	67
4.2.1	Data preparation	67
4.2.2	Structure overview	68
4.2.3	Training stages	70
4.3	Experiment	73
4.4	Results and discussion	73
4.5	Summary and conclusion	76
Chapter 5	Multi-Label Image Classification	78
5.1	Introduction	78
5.1.1	Contributions	80
5.2	Proposed Xclassifier model	81
5.2.1	Data preparation	81
5.2.2	Multi-label classifier	81

5.3	Experiment	85
5.4	Results and discussion	86
5.5	Summary and conclusion	89
Chapter 6 Multi-Class Image Classification: COVID-19 Detection		90
6.1	Introduction	90
6.1.1	Contributions	95
6.2	Proposed CovidXrayNet model	95
6.2.1	Proposed COVIDcxr dataset	95
6.2.2	Data preparation	96
6.2.3	Architecture	96
6.2.3.1	Data augmentation	97
6.2.3.2	CNN architectures and hyperparameters	102
6.3	Experiment	105
6.4	Results and discussion	108
6.4.1	Quantitative evaluation	108
6.4.2	Qualitative evaluation	109
6.4.3	Optimization in deep learning	109
6.4.4	Limitation and future direction	111
6.5	Summary and conclusion	113
Chapter 7 Conclusion		115
7.1	Summary of the proposed methods	115
7.2	Summary of findings	116
7.3	Future work	117
Bibliography		120
Appendix A Poster		145
	MultiViewModel	145
	Xclassifier	147
Appendix B Jupyter Notebook		149

CONTENTS

xvii

CXRlabeler	149
MultiViewModel	167
Xclassifier	193
COVIDcxr	197
CovidXrayNet.....	209

List of Figures

1.1	A Radiologist Examines a CXR Image for Thorax and Lung Diseases.	2
1.2	Graphic Outline of Thesis.	6
2.1	Deep Learning Applications in Chest Radiography, as Discussed in this Literature Review.	9
2.2	Example of a Radiology Report and Associated Images (Obtained from the Indiana University X-ray dataset) [28].	10
2.3	Radiology Imaging Modalities and Characteristics. Note: X-ray (a), CT (b), MRI (c), US (d), and image characteristics (e) [28].	12
2.4	Deep Learning [28].	17
2.5	The CheXpert Labels Heatmap.	28
2.6	The MIMIC-CXR Labels Heatmap.	29
3.1	Example of a Labeled Report from the MIMIC-CXR Dataset. Each label contains one of four values, 1.0, -1.0 , 0.0, or NaN , which indicate positive, negative, uncertain, or missing observations, respectively [132].	55
3.2	The CXRlabeler Structure [132].	57
4.1	A Basic Residual Block [146].	66
4.2	Examples of 12 Thoracic Diseases from MIMIC-CXR Dataset. Each disease is associated with frontal and lateral views of CXRs [146].	68
4.3	Overall Illustration of MultiViewModel [146].	71
4.4	Fluctuated LR. Per pathology, the plot on the right represents the LR after the stage 1 training, and the plot on the left shows the LR after the stage 2 training. Note that the x-axis represents what happened as the LR increased, and the y-axis indicates what the loss was (color figure online) [146].	72

4.5	Examples of the Most Confused CXRs with Heatmaps. Each image was associated with the prediction, actual, loss and probability values after the stage 1 training, where 0 and 1 represent negative and positive pathology respectively [146].	77
5.1	The CXR Image Formats [152].	80
5.2	The Xclassifier Structure [152].	83
5.3	Visualizing Parallel Training Approaches. We used four Tesla V100 GPUs and trained DenseNetblur-121d for multi-label classification tasks [152].	85
5.4	Correct Output Sample by Xclassifier [152].	88
6.1	Dataset Distribution [119].	97
6.2	The CovidXrayNet Structure [119].	98
6.3	Visualizing Data Augmentation Effects on a CXR. The CXR is for a 25-year-old COVID-19-positive female taken from the COVID-19 Image Data Collection [119].	101
6.4	Resizing Method. We propose squishing a 480×480 pixel CXR rather than cropping it to preserve important CXR details at the edges of the image [119].	102
6.5	Top Prediction Errors Generated by CovidXrayNet on COVIDx Test Dataset [119].	110
6.6	Randomly Generated Results for CovidXrayNet on COVIDx Test Dataset [119].	112
6.7	Confusion Matrix for CovidXrayNet on COVIDx Test Dataset [119].	112
6.8	Data Loader from COVIDcxr that Combines both Tabular Data and CXRs [119].	113

List of Tables

2.1 Activation Function for DL.	19
2.2 The CNN Architectures (ILSVRC Winners).	20
2.3 The CXR Datasets Employed in this Thesis. Our dataset is bolded.	26
2.4 The COVID-19 CXR Datasets.	31
2.5 The CXR Report Labelers. Our contribution in this thesis is bolded.	33
2.6 The DL Models for Classifying CXR Images. Our contributions in this thesis are bolded.	36
2.7 The DL Models for Classifying Multiple Image Modalities.	41
2.8 The DL Models for Generating Sentence-Level Radiology Reports.	41
2.9 The DL Models for Generating Paragraph-Level Radiology Reports.	43
2.10 Evaluation Metrics (Binary Classification Measures).	46
2.11 Evaluation Metrics (Image Caption Measures).	50
2.12 Quantitative Evaluation of Generated Radiology Reports based on DL Models.	50
3.1 Frequency of the 14 Labels in the Preprocessed MIMIC-CXR Dataset. The study extracted 156,790 unique impressions, of which 152,855 were used for training and 3,935 for testing. It reported the number of positive and negative cases for each label, with missing and uncertain labels considered negative labels.	58
3.2 The Frequency of Nine English Labels in the Preprocessed PadChest Dataset. The study extracted 29,365 unique Spanish reports, of which 22,275 were for training and 7,090 for testing. The testing split was manually labeled [115].	59
3.3 Comparing CXRlabeler with the Benchmarks in Labeling CXR Reports. CXRlabeler classifies each label as positive or negative.	62

3.4 Examples of the Labels Predicted by CXRlabeler and the Target Labels.	63
4.1 The MIMIC-CXR Dataset with 12 Labeled Pathologies. We counted the numbers of positive and negative observations in 10% of the dataset.	69
4.2 Data Augmentation for the CXRs. We applied a list of transforms parameters to the trained images.	69
4.3 The AUC per Epoch for Training the ResNet-50 CNN. This model detects cardiomegaly using CXRs of 299×299 or 224×224 pixels of chest X-rays.	69
4.4 Time per Epoch for Training the ResNet-50 CNN. This model detects cardiomegaly using a single NVIDIA Tesla P4 GPU or four NVIDIA Tesla P4 Graphics Processing Units (GPUs) in a parallel training. Note that the batch size was set to 64 images, and the image size was set to 224 pixels.	74
4.5 The Compression of the AUC Scores in each Epoch. We trained each pathology for eight epochs.	74
4.6 The Compression of AUC Scores. The DualNet model used an older limited released version of the MIMIC-CXR dataset. Our model used 10% of the publicly released version of the dataset. Note that we ignored uncertain and unknown labels.	76
5.1 Positive Label Co-occurrence of MIMIC-CXR.	82
5.2 Positive Label Co-occurrence of CheXpert.	82
5.3 The DenseNet-121 Variations Models and Training Performances. We used the full MIMIC-CXR dataset and trained for 10 epochs.	84
5.4 Image Formats for the CXRs and Training Performance. We used 10% of the MIMIC-CXR and trained ResNet-18 for 10 epochs.	86
5.5 Training Approaches and Training Performance. We used the NVIDIA V100 GPU.	87
5.6 Comparing Xclassifier with the Benchmark.	88
6.1 Models for Detecting COVID-19 from CXRs.	93
6.2 Data Augmentation for Detecting COVID-19 from CXRs.	94
6.3 The CNN Hyperparameters for Detecting COVID-19 from CXRs.	94

6.4 Pipeline for Data Augmentation on CXRs. For each independent parameter, we trained ResNet-18 on COVIDcxr for 30 epochs to examine the effects of various transformers on COVID-19 CXR classification.	100
6.5 The CNN Architectures on COVIDx and COVIDcxr. We trained the popular CNN architectures on both datasets for 30 epochs using the optimized data augmentation pipeline.	103
6.6 Optimizing CNN Hyperparameters using COVIDcxr. For each independent parameter, we trained several architectures on COVIDcxr to examine the effects of various hyperparameters on the accuracy of COVID-19 CXR classification.	106
6.7 Optimizing CNN Hyperparameters using COVIDx. For each independent parameter, we trained several architectures on COVIDx to examine the effects of various hyperparameters on the accuracy of COVID-19 CXR classification.	107
6.8 Comparing our Optimized Data Augmentation Pipeline and CNN Hyperparameters with the Benchmark. Both papers used VGG-19 and ResNet-50 on the COVIDx dataset but with different transformers and hyperparameters.	108
6.9 Comparing CovidXrayNet with the Benchmark. All models were based on a three-class COVID-19 classification; COVID-Net and CovidXrayNet employed the COVIDx dataset.	109

Introduction

1.1 Background

Chest radiography has been a cornerstone of medical imaging for many decades and remains the most common radiological exam in the world, as a radiologist may need to read and report more than 100 Chest X-rays (CXRs) per day [1]. This demand for CXR images may be attributed to their reasonable sensitivity to a range of pathologies, combined with their low radiation dose and cost-effectiveness. Figure 1.1 shows a radiologist who is examining a CXR for thorax and lung diseases.

However, CXR interpretation can be challenging in terms of detecting small or subtle pathologies, distinguishing between abnormality patterns, or detecting pathologies in specific locations (such as detecting a nodule posterior to the heart in a frontal CXR, due to the projection direction and the superimposition of anatomical structures. As a result, since the 1960s, researchers have been interested in automated pathology detection systems for CXR images [2][3][4]. Early systems had their limitations due to the complexity of CXR interpretation.

Recently, Deep Learning (DL) has had a tremendous impact on CXR interpretation. This is a relatively recent field of study because publicly available CXR datasets have not been around for very long. The earliest available dataset has only been available since 2015, and it contains less than 10,000 CXRs. By DL standards, this is a very small dataset. A very commonly used dataset in DL is ImageNet, which was introduced in 2009 and contains 3.2 million images [5]. Between 2017 and 2022, nearly one million labeled CXRs were released to empower DL researchers.

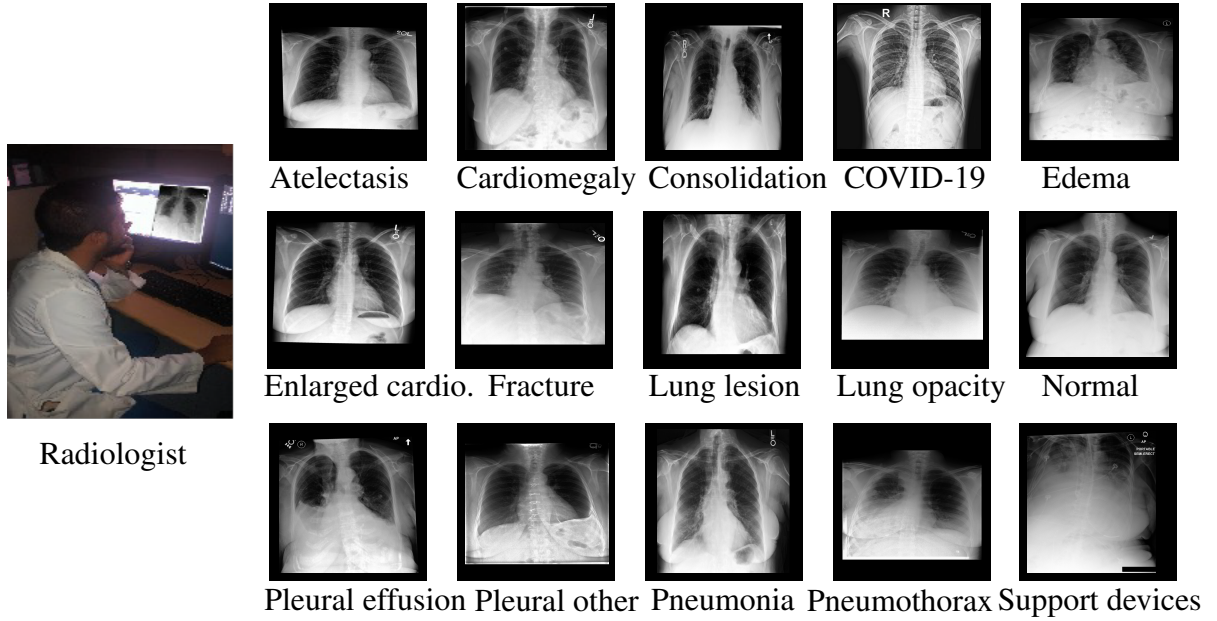


FIGURE 1.1. A Radiologist Examines a CXR Image for Thorax and Lung Diseases.

Deep learning is a promising subfield of machine learning, which in turn is a subfield of Artificial Intelligence (AI). DL algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations. This usually happens at multiple levels, with higher-level features being defined by lower-level features [6]. Basically, a linear combination of input signals, $x_1, x_2, x_3, \dots, x_m$, adds bias b_k to apply an affine transformation to generate the output, y_k (refer to Eq. (1.1)), where $W_{k1}, W_{k2}, W_{k3}, \dots, W_{km}$ are the weights and $\varphi(\cdot)$ is the activation function (e.g., rectified linear unit, ReLU [7]) [8]. The main computational element, named the "neuron" or "perceptron" enables DL machines to learn from experience without the need for expertise to specify the desired knowledge. Deep learning has already succeeded in many computerized applications including, but not limited to, computer vision, Natural Language Processing (NLP), speech processing, gaming, and cross-media retrieval [9].

$$y_k = \varphi\left(\sum_{j=1}^m W_{kj} X_j + b_k\right) \quad (1.1)$$

The most common approach to CXR interpretation is to use publicly available datasets to train DL models, with researchers using Convolutional Neural Network (CNN) for image analysis and developing Recurrent Neural Network (RNN) for NLP. While CNNs are a preferable networks that include the pixels in an image and other clear, spatially structured data, RNNs are good with natural language and similar, sequentially ordered data [10].

1.2 Motivation

Thorax diseases pose a serious risk to public health. Pneumonia, for example, affects about 450 million people (about 7% of the world's population) and causes nearly four million fatalities per year [11]. Additionally, the lung disease Coronavirus Disease 2019 (COVID-19) became a global pandemic in less than four months after first appearing in December 2019 in Wuhan, China [12]. A CXR may be examined by a radiologist to inspect any visual indicators linked to pneumonia, COVID-19, and several other diseases. Chest X-ray is the most common imaging examination performed worldwide, and it is essential for the screening, diagnosis, and treatment of many life-threatening diseases [13]. Currently, radiology images are interpreted by radiologists, who are limited by speed, fatigue, and experience. Certified radiologists are rare due to training costs. As a result, many healthcare systems outsource the task of medical image analysis. For example, there are teleradiology companies in India [14]. Additionally, a delay or errors in diagnosis can cause harm to a patient. Therefore, it is very important to develop CXR image classification methods to support radiologists.

Overall, the ultimate goal of this dissertation is to classify CXR images using DL due to the recent availability of large labeled CXR datasets, the complexity of their interpretation, and their value in clinical practice. We will investigate the current literature, extract labels from CXR reports, and classify CXRs based on binary, multi-label, and multi-class classification approaches. In brief, in the binary classification task, we classify each CXR into only one label out of two classes (i.e., positive or negative). In the multi-class classification, we classify each CXR into only one output class (e.g., COVID-19 vs. pneumonia vs. normal). However,

in the multi-label classification, each CXR can have multiple output classes (i.e., multiple pathologies).

This thesis will strive to address the following critical research questions:

- RQ 1. How can we contribute to the automatic interpretation of CXR research? What are the strengths and limitations of current DL-based models in this field? What studies are essential to fill the gaps?
- RQ 2. How can large CXR datasets be labeled automatically and accurately using radiology reports to aid in the training of deep neural networks?
- RQ 3. How can we classify the large volume of CXRs by integrating DL techniques?
- RQ 4. How can we make DL models more efficient in terms of computing resources and accuracy in detecting multiple diseases from CXRs?
- RQ 5. Which CNN optimization techniques can improve the accuracy of detecting critical diseases from CXRs?

1.3 Contributions

We investigate the DL path employed in radiology, from report labeling to image classification and image captioning. Our investigation includes more than 100 papers related to DL techniques and tailored to the radiology domain. This is an area of research that we anticipate will grow in the near future. As a result, our survey will be useful to researchers who want to apply DL to the radiology field. It will help them understand radiology text and image structures, apply DL algorithms, use available datasets, label reports, classify images, generate reports, and evaluate models. This **comprehensive survey** addresses **RQ 1.** and is presented in **Chapter 2** of this thesis and published as **J1**.

One of the primary challenges in the development of CXR classification models is labeling large datasets with multi-label image annotations extracted from radiology reports. Differing from rule-based methods, which cannot handle the extensive linguistic ambiguity in radiology reports, including misspellings and broken grammar, we propose a novel DL model named

CXRlabeler. It takes raw radiology text as the input and extracts multiple positive and negative CXR observations as the output. It utilizes the encoder learned from fine-tuning a Language Model (LM) on radiology reports to label these reports. To the best of our knowledge, CXRlabeler is the first proposed model that has the benefits of both LM fine-tuning and classifier fine-tuning to achieve highly accurate automated CXR report labeling. **CXRlabeler** addresses **RQ 2.** and is described in **Chapter 3** of this thesis and published as **C2.**

Consistent with recently proposed CNN models on automated CXR binary classification, we focus on training CNN models to detect common thoracic diseases. We propose a novel stage-wise training approach, named MultiViewModel, and observe the model's performance to reduce the training time and increase the accuracy. It is founded on a Residual Network (ResNet) architecture and a combination of recent techniques, including transfer learning, fine-tuning, fit-one-cycle functions, and discriminative learning rates. **MultiViewModel** addresses **RQ 3.** and is explained in **Chapter 4** of this thesis and published as **C1.**

Regarding the multi-label CXR classification task, we quantify the value of the optimal image format, demonstrate how distributed parallel DL accelerates neural network training, and compare the performances of variations of Densely Connected Convolutional Network (DenseNet). Xclassifier is an efficient multi-label classifier that trains an enhanced DenseNet-121 framework with blur pooling to detect multiple observations from a CXR. It uses the right amount of memory, runs on multiple graphics processing units, and has a high Area Under the Receiver Operating Characteristic Curve (AUC) on two large chest radiography datasets. **Xclassifier** addresses **RQ 4.** and is demonstrated in **Chapter 5** of this thesis and published as **C3.**

Our main contribution to the multi-class classification task is implementing CovidXrayNet, which classifies a CXR into either "COVID-19," "normal," or "pneumonia." It improves the detection rate of COVID-19 from CXRs by optimizing the data augmentation pipeline and CNN hyperparameters. To the best of our knowledge, CovidXrayNet is one of the first models to demonstrate the effects of data augmentation pipelines on CXR quality while also investigating several CNN hyperparameters. This, in turn, may significantly enhance the accuracy of CNN in diagnosing critical diseases. In addition, we introduce **COVIDcxr**, a

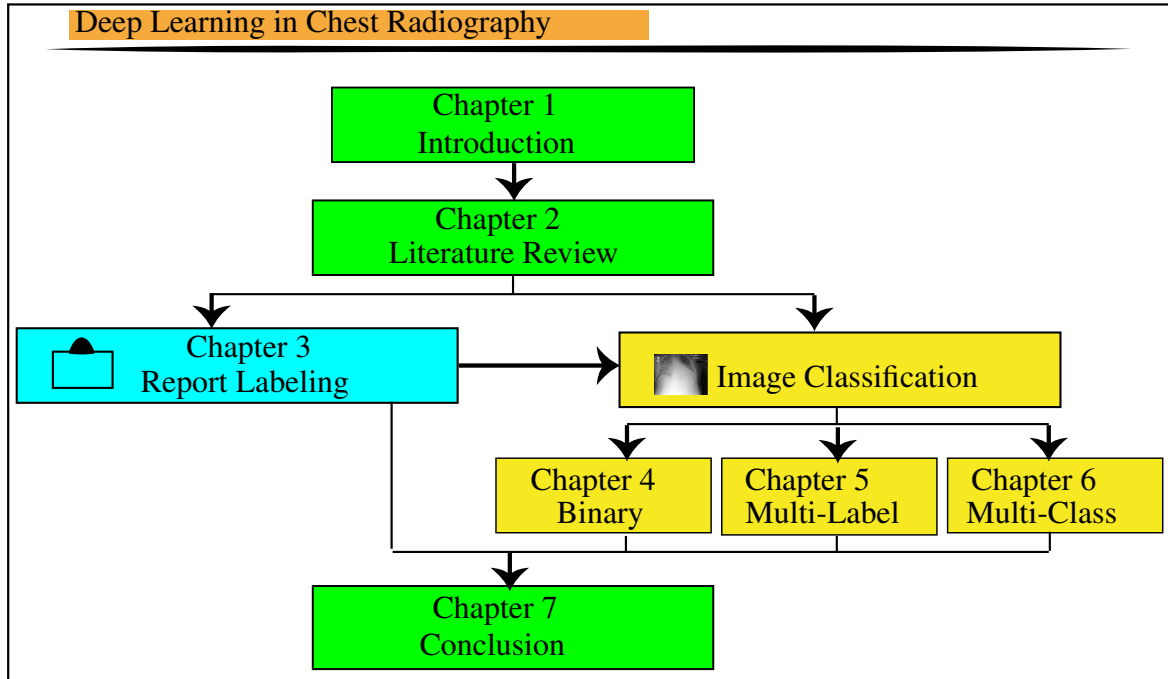


FIGURE 1.2. Graphic Outline of Thesis.

balanced and complete dataset that consists of 960 CXRs and the associated tabular data. **CovidXrayNet** addresses **RQ 5**, and is expounded in **Chapter 6** of this thesis and published as **J2**.

1.4 Outline

Figure 1.2 shows a graphic representation of this thesis's outline. The remainder of this thesis is structured as follows: **Chapter 2** provides a comprehensive review of DL in the radiology domain to describe trends and gaps in the field. In **Chapter 3**, our CXR report labeling tool is described in detail. The proposed model for binary classification in chest radiography is provided in **Chapter 4**. The multi-label CXR classification model is provided in **Chapter 5**. The multi-class classifier of COVID-19 is explained in **Chapter 6**. The thesis concludes in **Chapter 7**, with a brief summary of our contributions as well as an extensive discussion of our future research directions.

Literature Review¹

Substantial progress has been made toward implementing automated Chest X-ray (CXR) interpretation models based on Deep Learning (DL). This is due to the introduction of large labeled image datasets. In this chapter, we will investigate the following critical challenges: understanding radiology text and image structures, applying DL algorithms, utilizing available datasets, labeling reports, classifying images, generating reports, and evaluating models. We conclude the chapter with a critical discussion of these challenges and future research recommendations. This comprehensive survey will be useful for researchers interested in DL, particularly those interested in applying DL to the radiology domain.

2.1 Introduction

The combination of radiology images and text reports has led to research in the automatic interpretation of images, inspired by recent work in classifying and detecting objects and scenes for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [5]. Traditionally, Computer-Aided Detection (CAD) systems interpret medical images automatically to offer an objective diagnosis and assist radiologists [15]. Unlike CAD, DL is able to learn useful features that move beyond the limitations of radiology detection [16]. Researchers [17][18] have noted a significant performance increase in DL models over conventional CAD systems. For example, DL has been applied to mammography to discriminate between breast cancer and microcalcification [17], to ultrasounds to differentiate breast lesions (malignant and benign), and to Computed Tomography (CT) lung scans to classify pulmonary nodules

¹The content in this chapter has been published in *Artificial Intelligence in Medicine*, "Deep learning in generating radiology reports: A survey", Monshi, M., Poon, J., Chung, V. (2020).

[18]. Deep learning may help to improve patient safety by assisting radiologists in accurately interpreting CXRs, obtaining additional diagnostic criteria by generating unobservable data from imaging features, and increasing efficiency by performing various tasks automatically [19].

2.1.1 Contributions

In this chapter, we examine the DL path employed in radiology, from labeling reports to classifying images and generating reports. Unlike other recent surveys that have investigated DL in broad health informatics practices, our survey focuses exclusively on DL techniques tailored to the radiology domain, as shown in Fig. 2.1. Several recent surveys on DL applications [9][8] have been published in the literature on healthcare [20], electronic health records [10], health informatics [21], medical image analysis [14][22], and medicine [23][24]. However, no existing reviews specifically address image and text analysis, let alone that in radiology, at the time of writing this chapter. As such, this is the investigative scope of this literature review. Papers that cover a wide range of radiology applications and tasks based on DL have been analyzed. This is an area of research that we anticipate will grow in the near future.

2.2 Radiology

Radiology is a branch of medicine that can be divided into the following two subcategories: diagnostic and interventional [25]. Diagnostic radiologists examine medical images to diagnose the cause of a patient's symptoms, monitor treatment effects, screen for various illnesses, and then write radiology reports. However, interventional radiologists utilize radiology images to guide procedures. Currently, radiologists evaluate radiological images constrained by their speed, fatigue, and experience. Due to the high expense of training, certified radiologists are in short supply. As a result, many healthcare organizations delegate the work of medical image analysis to third parties. Teleradiology companies, for example, exist in India [14]. Furthermore, a patient may suffer pain due to a diagnosis delay or

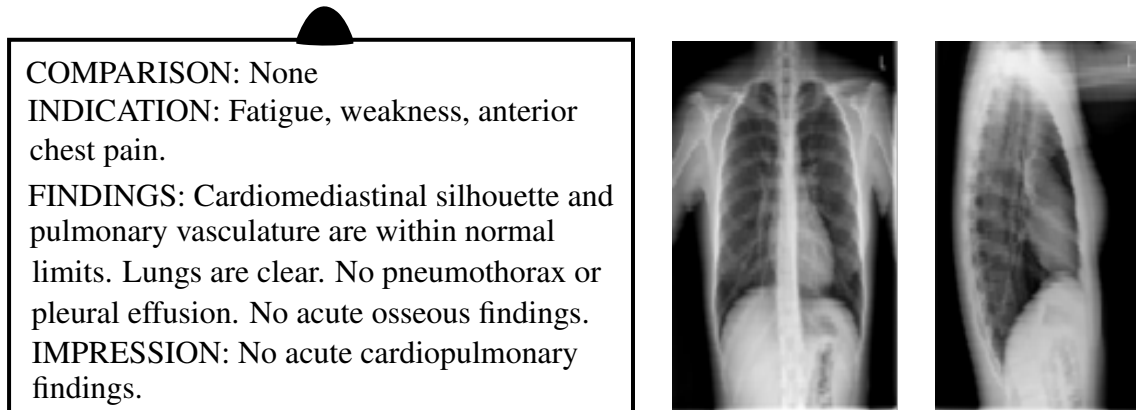


FIGURE 2.2. Example of a Radiology Report and Associated Images (Obtained from the Indiana University X-ray dataset) [28].

2.2.1 Understanding radiology text

A radiology report is a text-based document written by a certified radiologist. It contains descriptive information about a patient's history and symptoms and interpretations of relevant radiology images [26]. Normally, these reports are written in a specific radiology reporting format and divided into the following sections: comparison, indication, findings, and impressions. The impression section is the most crucial part of the report, as it describes medical observations of normal/abnormal features in order of perceived importance [27]. Figure 2.2 shows an example from the Indiana University Chest X-Ray (IU X-Ray) dataset [26]. Here, each report is associated with two CXR images.

There are several lexicons utilized in writing radiology reports, including Metathesaurus² [29], RadLex³ [30], and Medical Subject Headings (MeSH)⁴. Metathesaurus is a collection of more than five million concept names and a million biomedical terms from over 100 controlled vocabulary systems. In contrast, RadLex contains more radiology-specific terms than Metathesaurus, including those related to imaging methods and equipment. Furthermore, MeSH offers comprehensive controlled vocabulary created by the United States National Library of Medicine (NLM) to index scientific journal articles and books. Previously, Shin et.

²https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus.

³<https://www.rsna.org/practice-tools/data-tools-and-standards/radlex-radiology-lexicon>.

⁴<https://www.nlm.nih.gov/mesh/meshhome.html>.

al. [26] utilized MeSH terms to mine reports in IU X-Ray [31]. However, brain tumors and lung diseases do not have a fixed standardized lexicon. Instead, they have a semi-standardized description system.

The use of DL has shown promising results in generating radiology reports from images [27][32][33][34]. First, researchers generated short descriptive sentence of a radiology image using only the image features. Then, they attempted to produce more informative reports with multiple sentences. However, this introduced new challenges in content selection and ordering. Using this method, radiology reports can include information that cannot be detected from image features, such as the nationality of the patient [26]. Nevertheless, this text-based DL algorithm is insufficient as it does not include specific image labels.

2.2.2 Understanding radiology images

There are different types of radiology images, including X-ray, CT, Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), and Ultrasound (US) images [35]. Figure 2.3 shows examples of various radiology imaging modalities and characteristics.

Chest radiography is the most common imaging examination that demands correct and immediate interpretation to avoid life-threatening diseases [13]. A single radiologist may need to read and report more than 100 CXRs per day [1]. This imaging technology is starting to be employed as the first-line imaging modality by hospitals in Italy and the United Kingdom to diagnose patients with the Coronavirus Disease 2019 (COVID-19) [36]. Although CXR is less sensitive than chest CT, it is easy to document and may reduce the risk of cross-infection by utilizing portable radiology units [37].

Recently, several large CXR datasets were released to enable researchers to advance the state-of-the-art of the proposed DL models, as we will review in Section 2.4 (CXR Datasets). Consequently, CXRs have gained significant attention from DL researchers.

Picture Archiving and Communication Systems (PACSs) have been used since the 1990s by modern hospitals for radiology storage, management, transmission, and processing. They are e-system mainly used for the acquisition of medical images.



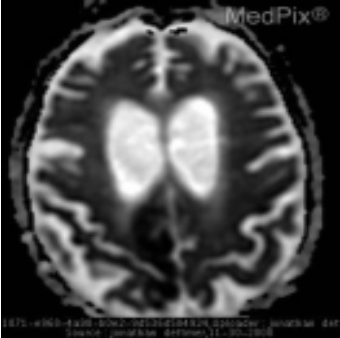

		Modality	Oragn	Size
	(a)	X-Ray	Chest	10 MB
	(b)	CT	Abdomen	250 MB
			Brain	150 MB
			Heart	1 GB
		MRI	Abdomen	50 MB
			Brain	100 MB
			Heart	240 MB
	(c)	PET	Brain	6 MB
			Heart	1 MB
	(d)	US	Heart	38 MB/s
	(e)			

FIGURE 2.3. Radiology Imaging Modalities and Characteristics. Note: X-ray (a), CT (b), MRI (c), US (d), and image characteristics (e) [28].

Digital Imaging and Communications in Medicine (DICOM) was introduced in 1993 to enhance standards and assist with many image processing procedures. It is the most common file format used to store medical imaging data for patient medical scans such as CXRs, CT scans, and MRI scans. It includes advanced report and result features [38].

The Joint Photographic Experts Group (JPEG) format, however, is utilized by most existing DL models in medical image prediction due to the limitations of Compute Engine machines. We will compare the DICOM and JPEG formats regarding the performance of multi-label classifiers for chest radiographs using DL in Chapter 5 (Multi-Label Image Classification).

From a DL perspective, radiology images are preprocessed differently due to varied processor and memory restrictions. Some images, such as X-rays, are Two-Dimensional (2D), while others, such as CT and MRI scans, are Three-Dimensional (3D). Currently, DL models that are trained on simple 2D images are more successful than those trained on 3D images, which

adds an extra dimension to the problem [39]. However, experience needs to be gained in applying DL to X-rays because they are 2D projections of a 3D human body [40]. In other words, DL algorithms may need to be adjusted to handle the physiological structures that lie on top of each other in X-rays. Significantly, DL algorithms, particularly Convolutional Neural Networks (CNNs), can process an input of 2D and 3D images with only minor adjustments. After all, DL in radiology images is still an area of active ongoing research.

So far, DL has been successfully applied to medical image analysis and acknowledged as a powerful tool for image classification [41], lesion detection [42], segmentation [43], content-based image retrieval [44], report generation from images, and image generation and enhancement [45]. To allow practitioners to rapidly implement DL solutions for image analysis tasks, NiftyNet⁵ [46] features an open source framework for many medical imaging CNN algorithms under the Apache License. Several surveys have introduced the role of DL algorithms in medical image analysis, focusing on CNNs [14][22]. Biswas et al. [47] classified DL models based on application area, including cardiovascular, neurology, mammography, microscopy, dermatology, gastroenterology, and pulmonary applications.

2.2.3 Understanding CXR findings

This section provides an overview of the classified findings in this thesis from CXR reports and CXR images. These are linked to the heart, lungs, and bones and include the following findings: atelectasis, cardiomegaly, consolidation, COVID-19, edema, enlarged cardiomediastinum, fracture, lung lesion, lung opacity, no finding, pleural effusion, pleural other, pneumonia, pneumothorax, and support device.

Atelectasis is a condition where a lung or part of a lung fails to expand completely [48]. It may be caused by pneumonia, pleural fluid, or lymph nodes compressing the bronchus, resulting in the closure of the lung. In an intensive care unit setup, atelectasis is the main cause of radio-opacity on a CXR image [49].

⁵<https://niftynet.io>.

Cardiomegaly is defined as enlargement of the heart [49]. It may be caused by human immunodeficiency virus, diabetes, pregnancy, kidney-related diseases, heart valve diseases, or thyroid disorders. However, it is frequently congenital [50]. The cardiothoracic ratio can be determined from a CXR to diagnose cardiomegaly wherein the heart is over 50% larger than the rib cage.

Consolidation is a condition in which a region of the lung contains liquid instead of air [51]. Swelling and hardening of a normal lung are often associated with consolidation [52]. Consolidation is an important CXR finding in several pathologies, especially pneumonia [53].

COVID-19 is an infectious disease caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), which first appeared in Wuhan, China, in December 2019 [54]. Then, it became a global pandemic on March 11, 2020. As of October 13, 2022, it has caused over 620.30 million cases and over 6.54 million fatalities globally [12]. Slowing the spread of this pandemic could be achieved by the early detection of positive COVID-19 cases from CXR images [55].

Edema occurs when excess fluids accumulate in a lung, resulting in breathing difficulties [56]. It may be caused by an acute lung injury or congestive heart failure. It appears on CXRs as opacities in the lungs, thickening of the bronchial walls, and hazy blood vessel contours. Detecting edema on CXRs is critical for treating patients with congestive heart failure [57].

Enlarged cardiomeastinum is mostly caused by cardiomegaly and refers to an enlarged heart [58]. Early detection of enlarged cardiomeastinum is advantageous for treatment, which may involve medications, medical procedures, or surgery.

Fracture represents inconsistent abnormalities in a CXR image, such as gross rib fracture [59]. Rib fractures are a common consequence of other injuries and can result in life-threatening complications [60]. Livingston [61] stated that nearly half of all rib fractures are missed in CXRs. Patients with these fractures may have increased morbidity and mortality rates.

Lung lesions might be an indication of severe diseases, such as lung cancer, heart diseases and respiratory diseases [62]. They are found on a spectrum ranging from focal to diffuse or

multifocal [59]. As lung cancer is the leading cause of cancer-associated deaths among men, it should be detected as early as possible to save patients' lives.

Lung opacity is an infectious condition that results in pneumonia [63]. It spreads to the lung region, causing suffocation, coughing, and fever. Pneumonia often presents in CXRs as a region of increased blackness and opacity in the lungs.

No finding is an observation used to capture the absence of the following 13 findings in a CXR image: atelectasis, cardiomegaly, consolidation, COVID-19, edema, enlarged cardiomeastinum, fracture, lung lesion, lung opacity, no finding, pleural effusion, pleural other, pneumonia, and pneumothorax [13]. However, it does not indicate that the CXR image is normal, as other pathologies may be present.

Pleural effusion, or effusion, is a disorder in which excess fluid accumulates between the chest wall and lungs. This extra fluid keeps the lungs from expanding, which can make it hard to breathe [50]. In critically ill patients, severe pleural effusion may contribute to hypoxemia during mechanical ventilation or cause tamponade physiology.

Pleural other is a label used to capture all abnormalities related to the pleural cavity [59]. When there are no sufficient samples of a pattern that involves the pleural cavity, "pleural other" is used to label a CXR image.

Pneumonia is a diagnosis that affects the alveoli due to inflammation in the lungs. "Pneumonia" was included as a label to represent the CXRs that suggested primary infection as the diagnosis, despite being a clinical diagnosis. [13]. Pneumonia affects nearly 7% of the world's population and causes about 4 million deaths annually, posing a severe risk to public health [11].

Pneumothorax occurs when air leaks from the lungs into the chest wall [60]. Typically, it is small and detected in the crowded lung apex area [59]. According to earlier research, 7.4-18 out of 100,000 males and 1.2-6 out of 100,000 women in America have pneumothorax each year [64]. Timely and correct diagnosis based on CXR is the key to successful treatment; otherwise, it can be fatal.

Support device is a label that indicates that a CXR image was taken using support devices such as chest tubes, central lines, endotracheal tubes, or nasogastric tubes. It is important to recognize the presence of a device in a CXR. A CXR image can be labeled as both "support device" and "no finding" [59].

2.3 Deep learning

Deep learning is a promising sub-field of Machine Learning (ML) which, in turn, is a sub-field of Artificial Intelligence (AI) (Fig. 2.4a). It occurs when a machine is composed of multiple layers, uses raw data as input, and improves the representations required for pattern recognition [6]. Essentially, a linear combination, V_k , of input signals, $x_1, x_2, x_3, \dots, x_m$, adds bias, b_k , to apply an affine transformation and generate the output, y_k (Fig. 2.4b), where, $w_{k1}, w_{k2}, w_{k3}, \dots, w_{km}$ are the weights, and $\varphi(\cdot)$ is the activation function (described in section 2.3.1). This main computational element, known as the "neuron" or "perceptron," enables the DL machine to learn from experience without the need to specify the desired knowledge.

Currently, DL has already succeeded in many computerized applications including computer vision, Natural Language Processing (NLP), speech processing, gaming, and cross-media retrieval. Additionally, DL models can be fed with multiple datatypes and iteratively distort them as they flow from layer to layer [20] (Fig. 2.4c). This is a particularly relevant function for radiology data, which consists of reports and linked images.

Researchers have classified DL models into three categories: supervised, unsupervised, and Reinforcement Learning (RL) [8][10]. Supervised learning, such as multilayer perceptron, Recurrent Neural Network (RNN), or CNN, infers a mapping function $f(x) = y$ from input x to output y . Recurrent neural networks have become a popular choice for mining radiology text to extract labels; CNNs, however, have gained popularity in radiology image classification. Additionally, RNNs can be accompanied with CNNs to generate medical image descriptions [26][34][65][66] (Fig. 2.4d). In contrast, unsupervised DL takes on board remarkable properties related to the distribution of x , including Boltzmann machines and

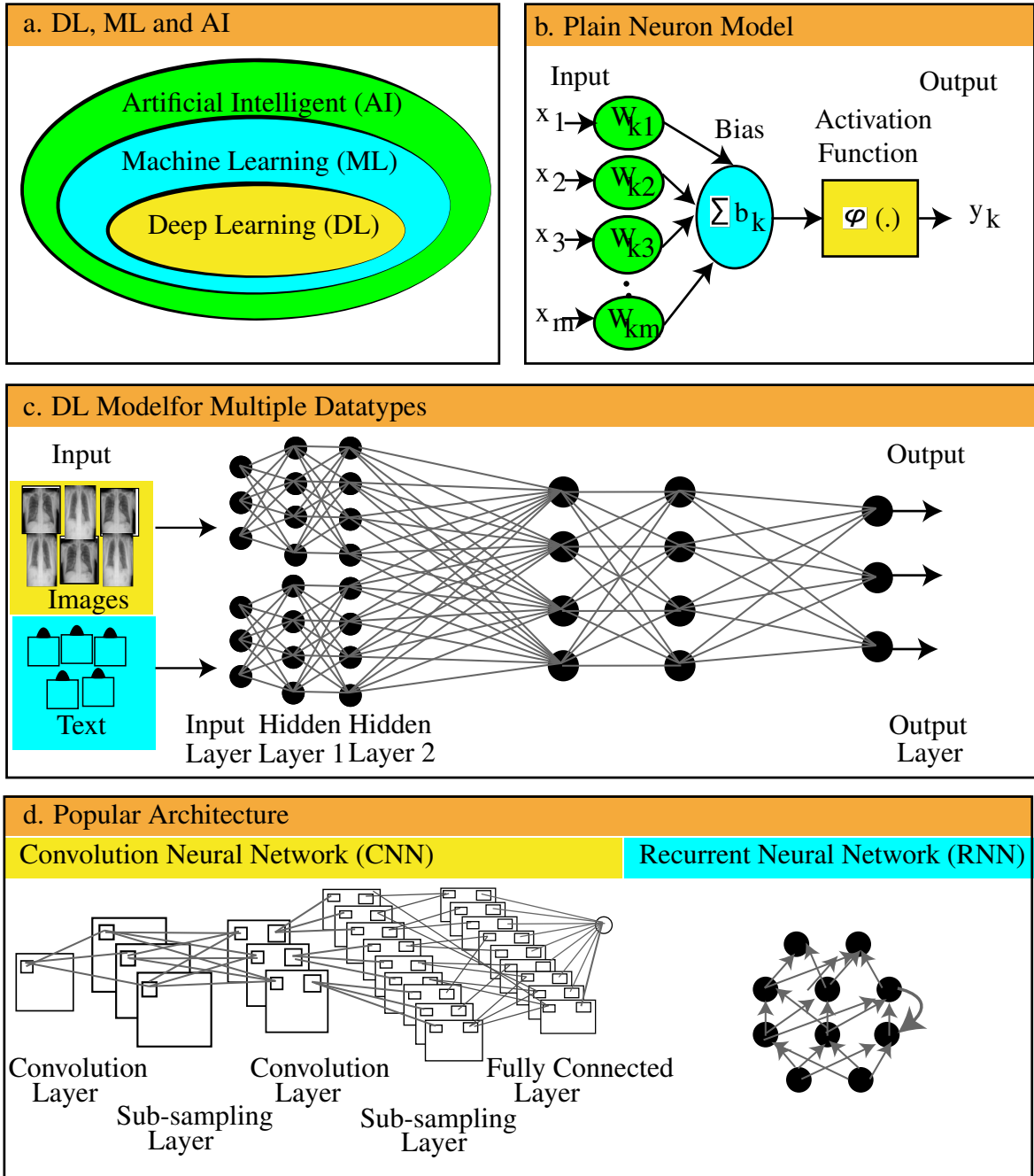


FIGURE 2.4. Deep Learning [28].

autoencoders. Deep RL is a semi-supervised technique for partially labeled datasets as it can act with limited input data. For instance, if a deep RL network is fed several tumor cells, it can overinterpret an image to detect insignificant aspects [67].

2.3.1 Activation function

An activation function is a critical element of DL as it adds non-linearity by taking the weighted sum of the inputs in one layer and converting it into an output value [68]. Then, this value is conveyed to nodes in the subsequent layer. Table 2.1 illustrates common activation functions including sigmoid, Hyperbolic Tangent (TanH), Rectified Linear Unit (ReLU) [7], and leaky ReLU functions [69]. The sigmoid function is one of the earliest activation methods used in neural networks but can cause network instability or freeze network learning. The limitations of the TanH function are similar, as it is a scaled form of the sigmoid function.

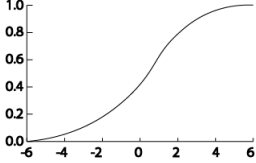
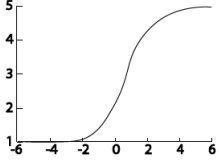
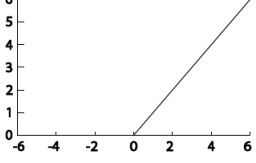
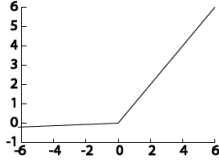
Meanwhile, the ReLU function performs better than sigmoid functions, as it was the first to be successfully used for neural networks by Glorot et. al., [7]. It converts the weighted sum of inputs to zero if they are less than zero or to the same inputs if they are equal to or greater than zero. The leaky ReLU function is an extension of the ReLU function that outputs small negative numbers if the inputs are negative. If not, it produces the same outputs as the ReLU function. Researchers tend to begin with ReLU functions and then apply other activation functions if they do not obtain optimal results.

All traditional CNN activation functions output a single result for a single input, except the softmax function. Instead, it produces multiple outputs. It is useful as it converts the output of the last neural network layer into a probability distribution. In practice, the softmax function is used in multi-class classifications, while the sigmoid function is used in binary classifications [70].

2.3.2 Convolutional neural network

A CNN is a type of multi-layer neural network that uses minimal processing to recognize visual patterns from pixel images [71]. One of the main advantages of a CNN is its ability to automatically amalgamate low-level features (including lines and edges) into high-level features (such as shapes) within subsequent layers [14]. For each, convolutional layer l , a set of k kernels, W_1, W_2, \dots, W_k , with biases b_1, b_2, \dots, b_k convolve an input image to generate feature maps, X_k . These generated maps have a non-linear transform, $\varphi(\cdot)$, in each layer

TABLE 2.1. Activation Function for DL.

Name	Equation	Plot	Characteristics
Sigmoid	$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$		Range [0,1] Not zero-centered Have an exponential center
TanH	$\text{tanh}(x) = \frac{2}{1+e^{-2x}} - 1$		Range [-1, 1] Zero-centered
ReLU [7]	$\text{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$		Does not saturate Fast
Leaky ReLU [69]	$\text{leakyReLU}(x) = \begin{cases} x, & x < 0 \\ \partial x, & x \geq 0 \end{cases}$		Overcomes the dead ReLU problem

(refer to Eq. [(2.1)]).

$$X_k^l = \varphi(W_k^{l-1} \times X^{l-1} + b_k^{l-1}) \quad (2.1)$$

There are several CNN models, including deep feed-forward CNNs for images and word-embedding networks for text. The histogram of oriented gradients and scale-invariant feature transform are two examples of convolutional image features. However, deep CNNs significantly outperform shallow learning frameworks and hand-crafted image features, as they need larger collections of training data [72].

Recently, CNNs have become the primary frameworks for mining medical data as the number of papers published on CNNs methods and applications has increased rapidly since 2015 [14][22]. In radiology, a CNNs is the most applicable DL algorithm for performing various tasks including medical image classification and segmentation [73]. Interestingly, CNNs can transfer learning from a large database unrelated to the current task (e.g., ImageNet [5]) into a related one (e.g., IU X-Ray [31]).

2.3.2.1 Architecture

The most popular CNN architectures have been proposed by top competitors at the ILSVRC. These include the following architectures: AlexNet [74], ZFNet [75], Visual Geometry Group (VGG) [76], GoogLeNet [77], Residual Network (ResNet) [78], ResNeXt [79], CUImage Team [80], and Squeeze and Excitation Network (SENet) [81], as listed in Table 2.2. ImageNet is a project that aims to create an enormous visual database that can be utilized by researchers in the field of visual object recognition [5]. It should be noted that ImageNet runs ILSVRC, an annual contest where software programmers classify and detect objects and scenes.

TABLE 2.2. The CNN Architectures (ILSVRC Winners).

Winer by year	No. of conv. layers	Top-5 error rate (%)
2012 - AlexNet [74]	8	16.4
2013 - ZFNet [75]	8	11.7
2014 second - VGG-16 [76]	16	7.4
2014 first - GoogLeNet [77]	22	6.67
2015 - ResNet [78]	152	3.57
2016 second - ResNeXt [79]	101	3.03
2016 first - CUImage Team [80]	152	2.99
2017 - SENet [81]	152	2.25

In 2012, Krizhevsky et al. [74] noted how AlexNet was the first model to considerably improve image classification performance. It obtained a 16.4% error rate using the ImageNet dataset. This model minimized the overfitting problem using data augmentation and dropout procedures. Two remarkable models were then proposed in 2014: VGG-16 (7.4% error rate), which reduces the spatial size of the input in each layer, and GoogLeNet (6.67% error

rate), which permits procedures, such as pooling and convolution to run in parallel to each other. AlexNet uses eight convolutional layers and 650,000 neurons (60,000,000 parameters) and has an error rate of 16.4%. In contrast, VGG-16 consists of 16 convolutional layers and 133,000,000 parameters and has a 7.4% error rates [82]. It is clear that VGG-16 is a significantly deeper model than AlexNet, which is why its error rate is lower.

By 2015, automatic image classification models could outperform human manual annotation with a 5%–10% error rate. This first occurred when He et al. [78] introduced Microsoft Deep ResNet. It contains 152 layers that apply residual connections in CNNs to address the issues of vanishing gradients [83] and degradation. The ILSVRC 2016 winner was the CUImage team [80], which assembled the following six architectures: Inception v3, Inception v4, Inception ResNet v2, ResNet 200, Wide ResNet 68, and Wide ResNet 3. However, the 2016 runner-up, ResNext [79], introduced a simple framework that consists of branches in a residual block. Each branch conducts a transformation aggregated by a summation function at the end. Although this model is based on ResNet and uses less layers, it outperforms ResNet, Inception-v3 and Inception ResNet v2 [84]. It can be generalizable by reshaping it using other models like AlexNet.

In 2017, the ILSVRC was concluded, as researchers considered the problem of supervised image classification solved [9]. The 2017 winner was SENets. These networks are based on the ResNeXt-152 model and adds recalibration to adaptively reweight feature maps.

To interpret radiology images, researchers follow some ImageNet CNN network settings as well as those of other reliable architectures. These include the Network in Network (NIN) [85] and Densely Connected Convolutional Network (DenseNet) [86], with slight modifications. For instance, Shin et al. [26], noted that AlexNet is a complex method; instead, they used NIN as it is a simpler and faster model. In addition, they suggested that GoogLeNet is the baseline CNN model, and they used it to train their data. Although AlexNet and GoogLeNet have different depths, Wang et al. [72] utilized both to train their Looped Deep Pseudo-Task Optimization (LDPO) network model.

When extracting features from images, VGG-16 is the preferred choice for some researchers in the visual pattern recognition community [87]. This is largely because VGG-16 offers uniform CNN architecture and publicly available weight configuration.⁶ For example, Shin et al. [65] and Dong et al. [66] adopted this architecture to read radiology images, and Yarnal [88] used it to classify CXRs.

2.3.3 Recurrent neural networks

An RNN is a neural network that processes sequential information while maintaining a state vector within its hidden neurons [89]. Equation (2.2) is the basic RNN that preserves hidden state h at time t , which is the outcome of a non-linear mapping from its input, x_t , and the previous state, h_{t-1} , where W and R are the shared weight matrices over time. Unlike CNNs, which are the preferred networks for pixels in an image and other clear, spatially structured data, RNNs work well with natural language and similar, sequentially ordered data [10]. They can predict the next words based on the former ones in a Language Model (LM) [90]. However, it is hard to save information for a long time, as the weights are equal in all RNN layers. Another issue is the requirement for a backpropagation algorithm to train the RNN as the gradients either grow or shrink. Consequently, variations of RNNs have been introduced to overcome these limitations.

$$h_t = \varphi(Wx_t + Rh_{t-1} + b) \quad (2.2)$$

The most popular extensions of RNNs are Long Short-Term Memory (LSTM) [91] and Gated Recurrent Unit (GRU) [92] architectures. Long short-term memory uses memory blocks to save the network temporal state and gates to monitor the information flow. Meanwhile, a GRU is a lighter form of RNN than LSTM in terms of topology, computational expenses, and complexity. At present, researchers must choose between the faster model offered by GRU that needs fewer parameters or the higher-performing model provided by LSTM that contains sufficient data and computational power [8].

⁶https://www.robots.ox.ac.uk/~vgg/research/very_deep/.

2.3.4 Data augmentation

A method that artificially inflates the original training set, S , with label-preserving transformations is data augmentation. It can be mapped as $\phi : S \mapsto T$, where T is the augmented set of S . The label-preserving transformation means that if image $x \subset y$, then $\phi(x) \subset y$ [93]. Hence, the artificially enlarged training set is defined as $S' = S \cup T$, where S' consists of S and the corresponding transformations denoted by ϕ . Resizing, flipping, and zooming are examples of data augmentation methods.

Data augmentation improves CNN performance [94], prevents overfitting [93], and is easy to implement [95]. Training a CNN on limited data inhibits its ability to generalize results to unseen data due to the over-fitting issue. However, inflating the dataset using data augmentation methods adds more invariant cases and thus prevents overfitting. In addition, generic methods are easy to implement and computationally inexpensive. Several recent works have proven the benefits of data augmentation in improving CNN-based models for various DL applications [93][94][95].

However, limited existing methods specifically address data augmentation in detecting diseases from CXRs. A shortcoming of existing studies is the limited amount of data augmentation methods evaluated. As such, we will investigate data augmentation briefly in Chapter 4 (Binary Image Classification) and thoroughly in Chapter 6 (Multi-Class Image Classification: COVID-19 Detection) because data augmentation leads to positive results when training CNN on limited data but only when using suitable augmentation techniques for each dataset [96].

2.3.5 Hyperparameters

Convolutional neural network hyperparameter optimization aims to find the optimal combination of values that must be selected for a given dataset before the training starts within a reasonable amount of time. The optimizer function, learning rate, loss function, number of epochs, and batch size are examples of CNN hyperparameters.

Deep learning practitioners aim to identify such values through automatic software, such as Optuna [97], or through a trial-and-error method. For example, Nishio et al. [98] utilized Optuna to implement Bayesian optimization in segmenting the lungs from severely abnormal CXRs. In Chapter 6 (Multi-Class Image Classification: COVID-19 Detection), we demonstrate the effects of hyperparameter optimization in diagnosing COVID-19 from CXRs.

2.3.6 Software

There are several software packages that support CNN and RNN implementations, including TensorFlow⁷ [99], Tensorpack⁸ [100], Keras⁹ [101], Convolutional Architecture for Fast Feature Embedding (Caffe)¹⁰ [102], PyTorch¹¹ [103][104], and fastai¹² [105]. These software packages are open_source projects that utilize NVIDIA support to enhance performance through Graphics Processing Unit (GPU) acceleration. Of note, DL training can be accelerated through an advanced GPU that facilitates parallel processing.

Using both TensorFlow and Tensorpack, Wang et al. [34] implemented a Text–Image Embedding Network (TieNet) that produces thorax disease reports. The DualNet [106] and the Hybrid Retrieval-Generation Reinforced Agent (HRGR-Agent) [27] frameworks are based on PyTorch. Caffe is a common software package utilized by practitioners to classify multiple image modalities. Using Caffe, Shin et al. [65] trained their deep CNN model to map X-rays into specified document categories, and Kisilev et al. [107] implemented a multi-task-loss CNN model to describe medical images. Additionally, using Caffe, Dong et al. [66], Wang et al. [72], and Rajpurkar et al. [108] acquired pre-trained CNN models on ImageNet for their radiology annotation systems.

⁷<https://www.tensorflow.org/>.

⁸<https://github.com/ppwyyxx/tensorpack/>.

⁹<https://keras.io>.

¹⁰<http://caffe.berkeleyvision.org/>.

¹¹<http://pytorch.org/>.

¹²<https://docs.fast.ai>

PyTorch and fastai are the main software used to propose various models in this thesis. They simplify training fast and accurate neural nets using modern best practices [109]. We also use fastai's low-level flexibility and optional high-level convenience to work on different research ideas while shortening training cycles.

2.4 CXR datasets

This section introduces employed CXR datasets in this thesis, as summarized in Table 2.3, and the COVID-19 CXR datasets, as presented in Table 2.4. Then, briefly, we outline some private CXR datasets and public datasets of different medical image modalities. Researchers have employed these radiology datasets for developing and evaluating DL models.

2.4.1 IU X-Ray

The IU X-Ray dataset [31] was the first large X-ray dataset that was released in 2015. It consists of 7,470 CXRs with 3,955 radiology reports available through OpenI; OpenI is an open-source collection of literature and biomedical images. It contains IU X-Ray, 2,064 orthopedic illustrations, and more than three million images from PubMed and the NLM. Often researchers refers to IU X-Ray as "OpenI".

However, the data in IU X-Ray comes from fully anonymized reports from two hospitals. As a result, some keywords, findings, and images are missing. Additionally, IU X-Ray is relatively small because the pathologies were manually annotated. Researchers [27][26][32][33][34] have used this dataset to demonstrate how their proposed DL models label and describe the diseases associated with the images.

TABLE 2.3. The CXR Datasets Employed in this Thesis. Our dataset is bolded.

Dataset	Source	Patients Count	Images Count	Images Format	Reports Count	Lables Count	Lables Method	Base annotation	Employed by
Open-I (IU X-ray) ¹³ Demner-Fushman, et al. [31] 2015	Indiana University	NOS	7,470	DICOM	3,955	NOS	Manual	Thorax diseases.	CXRlabeler (ch.3), [27][26][32][33][34]
ChestX-ray14 ¹⁴ Wang, et al. [110] 2017	National Institutes of Health (NIH)	30,805	112,120	PNG	private	14	NegBio	Consolidation, infiltration, pneumothorax, atelectasis, edema, emphysema, mass, fibrosis, effusion, pneumonia, nodule, pleural thickening, cardiomegaly & hernia.	CovidXrayNet (ch.6), [27] [34]
CheXpert ¹⁵ Irvin, et al. [13] 2019	Stanford Hospital	65,379	224,316	JPEG	private	14	CheXpert	Lung lesion, lung opacity, pneumothorax, pleural other, cardiomegaly, atelectasis, edema, enlarged cardiomeastinum, no finding, pleural effusion, pneumonia, fracture, consolidation & support devices.	Xclassifier (ch.5), [111][112]
MIMIC-CXR ¹⁶ Johnson, et al. [113] 2019	Beth Israel Deaconess Medical Center	65,079	377,110	JPEG (V1), DICOM (V2)	227,835	14	NegBio, CheXpert	Lung lesion, lung opacity, pneumothorax, pleural other, cardiomegaly, atelectasis, edema, enlarged cardiomeastinum, no finding, pleural effusion, pneumonia, fracture, consolidation & support devices.	CXRlabeler (ch.3), MultiViewModel (ch.4), Xclassifier (ch.5), [106][112][114]
PadChest ¹⁷ Bustos, et al. [115] 2019	Valencian Region Medical ImageBank (BIMCV)	67,625	168,861	DICOM	109,931	193	PadChest	Cardiomegaly, pleural effusion, pneumothorax, pneumonia, no finding, atelectasis, edema, consolidation, fracture & 184 more labels.	CXRlabeler (ch.3), [116]
COVID-19 image data collection ¹⁸ Cohen, et al. [117] 2020	Multiple	282	589	JPEG, PNG	0	20	NOS	Viral, bacterial, fungal, lipid, aspiration & unknown	CovidXrayNet (ch.6)
COVIDx ¹⁹ Wang, et al. [118] 2020	Multiple	13,870	13,975	JPEG, PNG	0	3	Auto	COVID-19, normal, & Pneumonia	CovidXrayNet (ch.6) [118]
COVIDcxr²⁰ Monshi, et al. [119] 2021	Multiple	960	960	JPEG, PNG	0	3	Auto	COVID-19, normal, & Pneumonia	CovidXrayNet (ch.6)

¹³<https://openi.nlm.nih.gov>.¹⁴<https://nihcc.app.box.com/v/ChestXray-NIHCC>.¹⁵<https://stanfordmlgroup.github.io/competitions/chexpert/>.¹⁶<https://archive.physionet.org/physiobank/database/mimiccxr/>.¹⁷<http://bimcv.cipf.es/bimcv-projects/padchest/>.¹⁸<https://github.com/ieee8023/covid-chestxray-dataset>.¹⁹<https://github.com/nbc-nlp/COVID-19-CT-CXR>.²⁰<https://github.com/MaramMonshi/CovidXrayNet/blob/main/Dataset/COVIDcxr-generate.ipynb>.

2.4.2 ChestX-ray14

The ChestX-ray14 dataset [110] (previously named ChestX-ray8) was released by the National Institute of Health (NIH) clinical center in 2017. At the time, this was the largest open access CXR dataset available, containing 112,120 frontal-view CXR images of 30,805 unique patients. However, the first CXR in this dataset was taken in 1992, and the most recent CXR was taken in 2015; X-ray technology evolved during this long time span.

At first, researchers used NLP to extract eight of the most common disease labels from the original radiological reports. Then, they used text mining to extract 14 prevalent diseases from the same radiological data to create the ChestX-ray14 labels. The labels are atelectasis, consolidation, infiltration, pneumothorax, edema, emphysema, fibrosis, effusion, pneumonia, pleural thickening, cardiomegaly, nodule, mass, and hernia. However, the complete text reports are not publicly available.

2.4.3 CheXpert

The CheXpert dataset [13] was made available by the Stanford Hospital in 2019. It consists of 224,316 chest radiographs of 65,240 patients. There are two variations of this dataset: a high-resolution dataset and a downsampled-resolution dataset.

Unlike ChestX-ray14, which uses an automatic labeler to extract labels from reports, CheXpert offers radiologists labeled validation and expert scores. Each radiograph is labeled with 14 observations: atelectasis, cardiomegaly, consolidation, edema, enlarged cardiomeastinum, fracture, lung lesion, lung opacity, no finding, pleural effusion, pleural other, pneumonia, pneumothorax, and support devices. The labels contain positive, negative, uncertain, and missing values. They were extracted from unstructured radiology reports using the CheXpert labeler. Figure 2.5 shows the dependencies between labels in the CheXpert dataset. For instance, pleural effusion CXRs are positively correlated with lung opacity CXRs.

Note that a CheXpert competition is organized by the Stanford Machine Learning Group, which maintains private testing data for final evaluation of the Area Under the Receiver

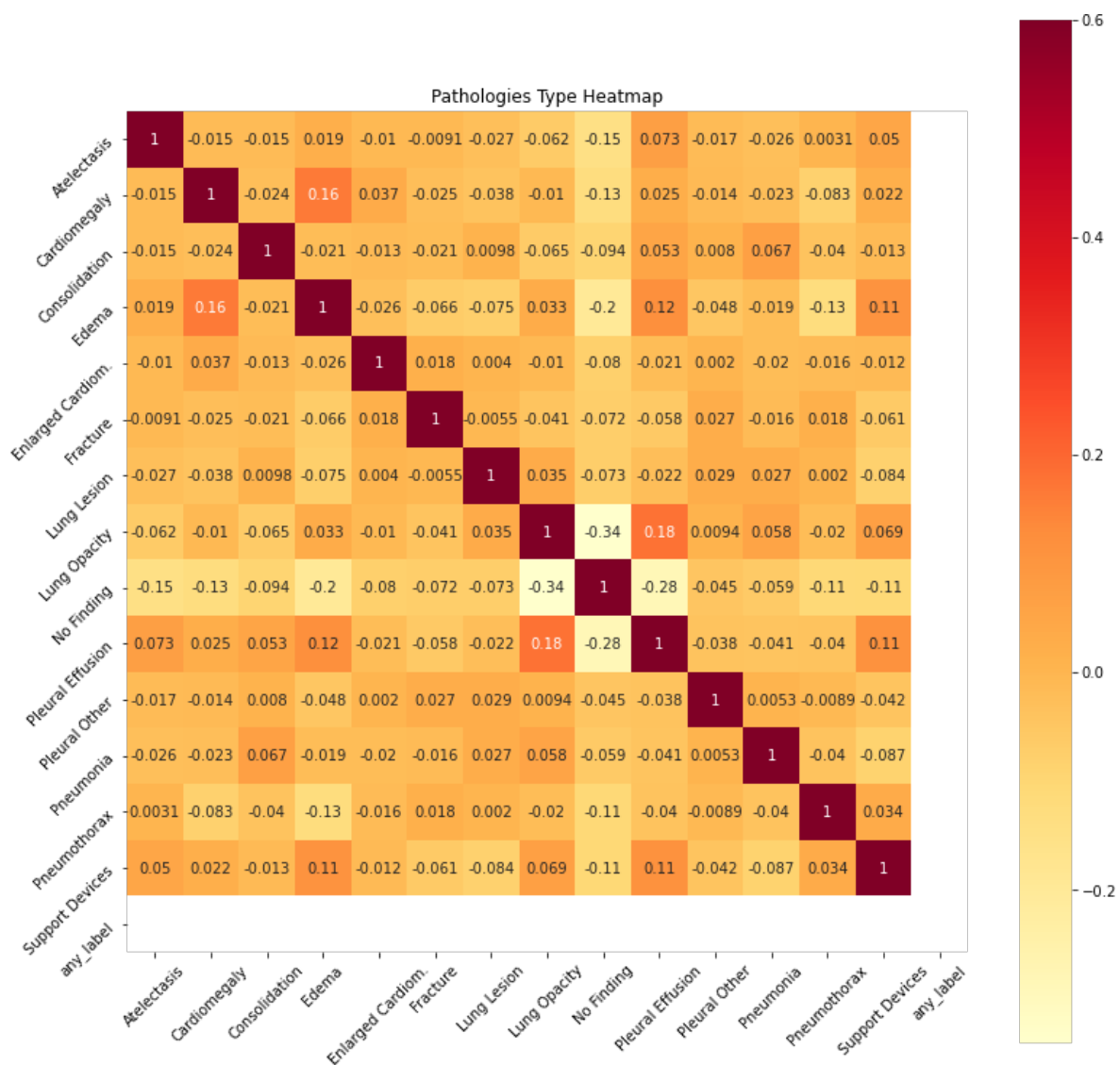


FIGURE 2.5. The CheXpert Labels Heatmap.

Operating Characteristic Curve (AUC) score on detecting five chosen diseases, i.e., atelectasis, cardiomegaly, edema, consolidation, and pleural effusion.

2.4.4 MIMIC-CXR

The Medical Information Mart for Intensive Care Chest X-ray (MIMIC-CXR) dataset [113] is the largest open access chest radiography dataset to date and was co-released with CheXpert by Beth Israel Deaconess Medical Center. It includes 377,110 CXRs linked to 227,835 reports

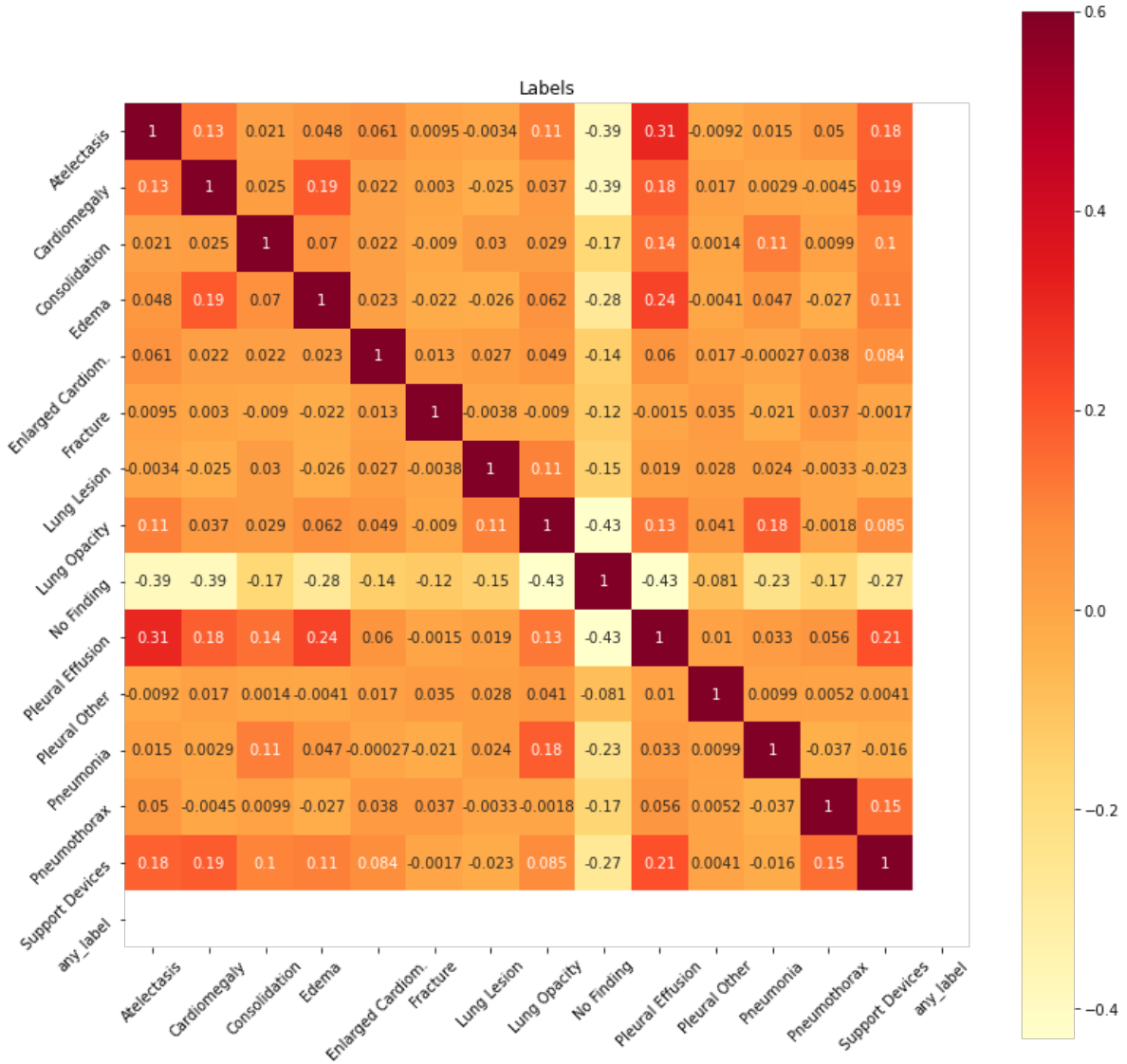


FIGURE 2.6. The MIMIC-CXR Labels Heatmap.

for 65,379 patients. There are two releases of this dataset, including the DICOM version [113] and the JPEG version [120]; the latter was generated by converting DICOM files into a more accessible format.

The MIMIC-CXR images were labeled by two automatic labelers, namely, the NegBio labeler [121] and CheXpert labeler [13]. Then, a board of experienced radiologists validated the generated labels against 687 reports and concluded that CheXpert outperformed NegBio. The labels include 12 pathologies (atelectasis, cardiomegaly, consolidation, edema, enlarged cardiome-diastinum, fracture, lung lesion, lung opacity, pleural effusion, pleural other, pneumonia,

and pneumothorax) as well as "support devices" and "no finding", which indicates the absence of all 12 pathologies. These labels overlap with those of the popular ChestX-ray14 dataset and match those of the co-released CheXpert dataset. Each label has four classes (positive, negative, uncertain, or missing). Figure 2.6 represents the label co-occurrence in this dataset. For instance, atelectasis CXRs are positively correlated with lung opacity CXRs.

2.4.5 PadChest

The PAtHology Detection in Chest radiographs (PadChest) dataset [115] contains 168,861 CXRs from six different views and the associated 109,931 Spanish reports of 67,625 patients, collected from San Juan Hospital. It provides researchers with the opportunity to address unfinished investigations, such as by measuring DL model performance using the CXR views [116].

Compared to other CXR datasets, PadChest is labeled with the largest number of English annotations including 174 radiology findings, 19 diagnoses, and 104 anatomic locations. These labels can be used regardless of the language because they are mapped onto the standard Unified Medical Language System [122].

2.4.6 COVID-19 datasets

Table 2.4 outlines the public datasets of COVID-19 CXRs. These datasets are constantly updated with new images added by researchers around the world. Nevertheless, none of these datasets provides complete metadata for all patients, except our proposed COVIDcxr dataset [119] (refer to section (6.2.1)).

The largest public dataset in terms of presented positive COVID-19 cases and the most popular dataset among researchers is COVIDx [118]. It includes 15,496 CXRs generated from five public datasets; where three of them— the COVID-19 Image Data Collection, Figure 1 COVID-19 Chest X-Ray Dataset Initiative, and ActualMed COVID-19 Chest X-Ray Dataset

Initiative can be downloaded from the GitHub repository, and two datasets—RSNA Pneumonia Detection Challenge dataset and COVID-19 Radiography Database can be obtained from Kaggle. It contains three classes of CXRs: COVID-19, pneumonia, and normal. Note that COVIDx is expanding on a regular basis with the addition of new patient records for training while maintaining the same test dataset for consistency.

However, COVIDx is unbalanced, as the number of cases in the COVID-19 class (589) is far lower than that in the pneumonia (6,056) and no_finding (8,851) classes. This may cause a sharp increase and decrease in the loss values while training a DL model. To address this issue, Bridge et al. [123] proposed a Generalized Extreme Value (GEV) as an alternative to the common sigmoid activation function. They proved that the GEV distribution improves the performance of COVID-19 classification from unbalanced datasets.

COVIDcxr is our proposed dataset in this thesis, aiming to create a balanced, unbiased, and complete COVID-19 CXR dataset. We will describe and use this dataset in Chapter 6 (Multi-Class Image Classification: COVID-19 Detection).

TABLE 2.4. The COVID-19 CXR Datasets.

Dataset	Description
Figure 1 COVID-19 Chest X-Ray Dataset Initiative ²¹ [124]	56 CXRs, metadata & clinical notes
ActualMed COVID-19 Chest X-Ray Dataset Initiative ²²	239 CXRs, metadata & clinical notes
covid-19-ct-cxr ²³ [125]	263 CXRs & relevant text
COVID-19 image data collection ²⁴ [117]	654 CXRs, metadata & clinical notes
COVID-19 radiography database ²⁵	219 COVID-19, 1341 normal & 1345 pneumonia CXRs
COVIDx ²⁶ [118]	13917 CXRs for training & 1579 CXRs for testing
COVIDcxr ²⁷ [119]	320 COVID-19, 320 normal & 320 pneumonia CXRs

²¹<https://github.com/agchung/figure1-COVID-chestxray-dataset>.

²²<https://github.com/agchung/Actualmed-COVID-chestxray-dataset>.

²³<https://github.com/ncbi-nlp/COVID-19-CT-CXR>.

²⁴<https://github.com/ieee8023/covid-chestxray-dataset>.

²⁵<https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.

²⁶<https://github.com/lindawangg/COVID-Net/blob/master/docs/COVIDx.md>.

²⁷<https://github.com/MaramMonshi/CovidXrayNet/blob/main/Dataset/COVIDcxr-generate.ipynb>.

2.4.7 Private datasets

Moreover, researchers have trained their DL frameworks on several privately-owned datasets, including the PACS from the NIH clinical center [65] and CX-CHR [27]. The PACS from the NIH clinical center consists of 216,000 2D images with radiology reports that offer visual references to pathologies. The CX-CHR dataset contains CXRs of 35,500 patients and contains Chinese reports.

2.4.8 Beyond CXR

Apart from X-ray collections, the Digital Database for Screening Mammography (DDSM)²⁸ [126] and the Pathology Education Informational Resource (PEIR)²⁹ [127] are open source datasets of different image modalities.

The PEIR is a digital library created by the University of Alabama for medical education. It contains 4,732 images and sentence-level descriptions of 20 different body parts, including the abdomen, adrenal, aorta, breast, chest, head, and kidneys.

The DDSM, however, contains 2,620 scanned films of normal, benign, and malignant breast mammography with verified pathology information. It is supported by the University of South Florida, and it has been widely used by researchers due to its scale and ground truth validation. Kisilev et al. [107] selected a subset of the DDSM database that consists of 974 images annotated with semantic descriptors to test their multi-task-loss CNN-based model. It outperformed the accuracy of current techniques by up to 10% when detecting and describing lesions.

2.5 CXR report labeling

Generally, radiology reports are semi-structured and use standardized documentation templates [113]. Consequently, researchers have proposed open_source NLP tools to extract

²⁸<http://www.eng.usf.edu/cvprg/Mammography/Database.html>.

²⁹<https://peir.path.uab.edu/library/>.

controlled vocabulary from radiology reports. Natural language processing explores the use of machines to process/understand human languages and carry out useful tasks. Traditional learning algorithms for NLP are often incapable of absorbing large volumes of training data, as feature engineering requires significant human expertise [128]. Several years ago, NLP was brought forward by a new era of DL algorithms using a philosophy named “NLP from scratch” [129]. Such DL waves have the capacity to learn representations from text through layers of nonlinear neurons for feature extraction.

Several NLP systems have been proposed for extracting medical labels from CXR reports. These are based on feature engineering, such as NegBio [121] and CheXpert [13], or DL algorithms, such as Recurrent Neural Network with Attention (RNN-ATT) [115], CheXpert++ [130], CheXbert [131], and our CXRlabeler [132] in Chapter 3. Table 2.5 summarizes these existing labelers.

TABLE 2.5. The CXR Report Labelers. Our contribution in this thesis is bolded.

Labeler	Model	Paper	Dataset
Feature Engineering	NegBio ³⁰	Peng et al. 2018 [121]	ChestX-ray14
	CheXpert ³¹	Irvin et al. 2019 [13]	CheXpert & MIMIC-CXR
Deep Learning	CheXpert++ ³²	McDermott et al. 2020 [130]	CheXpert
	CheXbert ³³	Smit et al. 2020 [131]	CheXpert
	RNN-ATT ³⁴	Bustos et al. 2020 [115]	PadChest
	CXRlabeler³⁵	Monshi et al. 2021 [132]	MIMIC-CXR, PadChest & IU x-ray

2.5.1 Feature engineering approaches

Feature engineering-based methods are rule-based systems that rely on medical terms and grammatical rules to extract structured labels from CXR reports. The most popular systems are the NegBio labeler and CheXpert labeler.

³⁰<https://github.com/ncbi-nlp/NegBio>.

³¹<https://github.com/stanfordmlgroup/chexpert-labeler>.

³²<https://github.com/mmcdermott/chexpertplusplus>.

³³<https://github.com/stanfordmlgroup/CheXbert>

³⁴<https://github.com/auriml/Rx-thorax-automatic-captioning>.

³⁵<https://github.com/MaramMonshi/CXRlabeler>.

NegBio was developed by the NIH and used to annotate the ChestX-ray14 dataset. It employs universal dependency and subgraph matching for pattern definition and graph traversal search, respectively. However, it extracts mentioned observations from the reports automatically using MetaMap [133] and DNorm [134], which may result in weak extraction as reported by Irvin et al. [13].

CheXpert was built by the Stanford Machine Learning Group and based on NegBio. It overcomes the limitations of NegBio by avoiding automatic mention extractors and capturing the variations of negation and uncertainty. As a result, CheXpert has achieved a higher F1 score. Although CheXpert is very useful in extracting thoracic labels, McDermott et al. [130] demonstrated three significant issues associated with its performance: a slow run-time speed, lack of differentiability, and lack of availability of continuous probabilistic output as it produces binary labels.

NegBio and CheXpert have been employed to generate labels for the most extensive publicly available CXR datasets, including ChestXray14, CheXpert, and MIMIC-CXR, despite their known limitations relating to label quality. For example, Oakden-Rayner et al. [135] estimated that the Positive Predictive Values (PPVs) of ChestXray14 labels are 10%_30% lower than the values noted in the clinical records.

2.5.2 Deep learning approaches

In contrast to the feature engineering_based approach to labeling radiology reports, DL-based methods are capable of capturing the complexity, ambiguity, and subtlety in the text. Therefore, recently, researchers have introduced CheXpert++, CheXbert, RNN-ATT, and this these introduces CXRlabeler.

CheXpert++ is based on Bidirectional Encoder Representations from Transformers (BERT) [136]. The CheXpert++ model is initialized from the clinical BERT [137] with a multitask classification head. It is a suitable drop-in replacement for the rule-based system, CheXpert, because it runs 1.8 times faster, generates better labels, and can be integrated with neural pipelines and active learning systems.

CheXbert is also based on BERT, but a more detailed annotation study was performed using two board-certified radiologists and error resolution policies, improving the labeling performance. However, CheXbert was fine-tuned with a small set of manual annotations augmented with automatic back_translation that introduced noise into the reports.

The **RNN-ATT** model combines a bi-directional LSTM and a per-label attention mechanism [138] in an RNN. As a result, it learns diverse text representations for each label. This model has achieved a 0.93 micro F1 score and was used to label 73% of the PadChest dataset. One downside is that the generated labels are not reliable when considering equal weights for each class because the macro F1 score is low (60.1%).

CXRlabeler is the method proposed in this thesis that combines the benefits of both LM fine-tuning and classifier fine-tuning to achieve highly accurate automated CXR report labeling. We have dedicated Chapter 3 (Report Labeling) to explaining and evaluating this model.

2.5.3 Beyond report labeling

Since 2010, DL has been productively applied to NLP tasks [139], including natural language generation from meaning representation. This can be considered the inverse of natural language understanding [140]. Through this, DL can generate fluent, communicative, and new image descriptions.

In addition to labeling reports, NLP assists with converting text into a structured report, extracting meaningful information when applied to a free-form radiologist text [141]. A recent NLP technique is neural language modeling, which includes word embedding and recurrent LMs [142]. Word embedding converts words into vectors to allow for less sparse data representation. Using this, DL models can be trained with smaller datasets. Advanced word embedding has been applied to a large collection of radiology reports to generate word vectors of radiology image descriptions [27][32][33][34][65][143]. Recurrent LMs predict word outputs based on a sequence of arbitrary past words. As such, they are not limited by fixed input dimensions.

2.6 CXR image classification

Binary, multi-class, and multi-label classifications are the most popular classification problems in radiology, as outlined in Table 2.6. In such tasks, the DL algorithms access the data labels, where data entries x_1, \dots, x_n have to be categorized into predefined classes C_1, \dots, C_l . If the input is to be classified into one of two non-overlapping classes (C_1, C_2), then this is a binary classification task. However, multi-class classification classifies the input into one of l non-overlapping classes. In addition, the input can be classified into several of l non-overlapping C_j in a multi-labeled classification task.

TABLE 2.6. The DL Models for Classifying CXR Images. Our contributions in this thesis are bolded.

Classifier	Model	Paper	Dataset
Binary	CheXNet	Rajpurkar et al. 2017 [108]	ChestX-ray14
	Learn-Diagnose	Wang et al. 2017 [144]	ChestX-ray14
	DualNet	Rubin et al. 2018 [106]	MIMIC-CXR
	Thorax-Net	Wang et al. 2019 [145]	ChestX-ray14
	MultiViewModel	Monshi et al. 2019 [146]	MIMIC-CXR
	AG-CNN	Guan et al. 2020 [147]	ChestX-ray14
	VGG16	Yarnall 2020 [88]	MIMIC-CXR
	COVIDX-Net	Hemdan 2020 [148]	COVID-19 collection
	CovXNet	Mahmud 2020 [149]	Guangzhou Medical Center & Sylhet Medical College
	ResNet-50	Narin 2020 [150]	COVID-19 collection & Kaggle
DarkCovidNet	Ozturk 2020 [151]	COVID-19 collection & ChestXray-14	
Multi-label	Latent-space	Gyawali et al. 2019 [111]	CheXpert
	CheXpert	Irvin et al. 2019 [13]	CheXpert
	CheXclusion	Seyyed-Kalantari et al. 2020 [112]	CheXpert & MIMIC-CXR
	VSE-GCN	Hou et al. 2021 [114]	MIMIC-CXR
	Xclassifier	Monshi et al. 2022 [152]	CheXpert & MIMIC-CXR
Multi-class	VGG-16	Nishio et al. 2020[153]	COVID-19 collection & RSNA
	DarkCovidNet	Ozturk 2020 [151]	COVID-19 collection & ChestXray-14
	CovXNet	Mahmud 2020 [149]	Guangzhou Medical Center & Sylhet Medical College
	COVID-Net	Wang et al. 2020 [118]	COVIDx
	MobileNet-v2	Apostolopoulos 2020 [154]	COVID-19 collection, RSNA, Radiopaedia, SIRM & Kermany
	CNN-SVM	Sethy et al. 2020 [155]	COVID-19 collection, COVID-19 radiography & Kermany
	CovidXrayNet	Monshi et al. 2021 [119]	COVIDx & COVIDx_r

2.6.1 Binary

The most common methods introduced to solve the CXR classification problem in the literature are based on binary classification with CNNs. Theoretically, binary classifiers are much less complicated than multi-label or multi-class classifiers because each CXR has only two

possible target outcomes (i.e., positive or negative). For instance, CheXNet [108], Learn-Diagnose [144], DualNet [106], Thorax-Net [145], our MultiViewModel [146], Attention-Guided Convolutional Neural Network (AG-CNN) [147], VGG-16 [88], COVIDX-Net [148] CovXNet, [149], ResNet-50 [150], and DarkCovidNet [151] train independent binary classifiers for each label with CNNs.

In 2017, CheXNet achieved benchmark performance in detecting pneumonia using a modified DenseNet. In 2019, Thorax-Net incorporated an additional attention branch into CNN based on gradient-weighted class activation mapping [156]. This exploited the correlation between labels and disease locations. In 2020, Yarnall [88] studied the effect of various CNN architectures with different hyperparameters on classification accuracy. The study used VGG-16 [76] with the ReLU activation function, resulting in an accuracy that ranged from 62.23% to 83.52% for each label. However, these single-label classifiers did not consider any pathology correlations and ignored the relationship information among labels.

Researchers tend to classify thorax diseases from frontal CXRs using ChestX-ray14 via neural networks such as CheXNet, Learn-Diagnose, Thorax-Net and AG-CNN. However, Bertrand et al. [116] and Hashir et al. [157] suggested that using lateral CXRs enhances the performance for certain prediction tasks, such as those for pleural effusion. Furthermore, Rubin et al. [106] proposed a DualNet model to prove that simultaneous processing of both frontal and lateral CXR inputs results in better classification performance. Unlike ChestX-ray14 [110], which only presents the frontal view of CXRs, MIMIC-CXR is a multi-view version of a radiograph dataset. DualNet employed a limited released version of the MIMIC-CXR dataset to automate the reading of frontal and lateral CXRs. In Chapter 4 (Binary Image Classification), we propose MultiViewModel to show that processing multi-view CXRs simultaneously results in better binary classification performance.

For a COVID-19 detection task, COVIDX-Net [148] achieved 90% with only 25 CXRs for patients with COVID-19 and 25 CXRs for normal patients. Another balanced dataset with 305 cases in each class was used to train the CovXNet model, resulting in 97.40% accuracy [149]. With a combination of ResNet-50, InceptionV3, and Inception-ResNetV2, Narin et al.'s [150] model achieved 98% accuracy. The 50 healthy CXRs in this study [150], however,

belonged to children (one to five years old) from a Kaggle repository [158]. Using a larger but unbalanced dataset of 1,125 images, DarkCovidNet [151] achieved 98.08% accuracy. Moreover, Wang et al. [159] localized the pulmonary location coordinates of COVID-19 (i.e., left lung, right lung, or both [bi-pulmonary]) using a residual attention network [160].

2.6.2 Multi-label

From a practical perspective, some CXR labels might be closely linked and their interdependency is very important for the final diagnostics. For example, infiltration is often associated with atelectasis [110], and cardiomegaly tends to be linked with pulmonary edema [144]. To examine multiple labels simultaneously, a latent-space self-ensemble model employed stacked semi-supervised learning using unsupervised disentangled representation learning [111]. This model achieved a 66.97% AUC on CheXpert [13].

Recently, the Visual-Semantic-Embedded Graph Convolutional Network (VSE-GCN) model fed joint features of label embeddings and visual features into a GCN to model the correlations among CXR labels [114]. Meanwhile, CheXclusion investigated fairness gaps in DL-based CXR classifiers to evaluate the true positive rate disparity for public datasets [112]. The VSE-GCN model and CheXclusion achieved 72.10% and 83.40% accuracy, respectively, on MIMIC-CXR [113]. In Chapter 5 (Multi-Label Image Classification), we extend this wave of research using more efficient training methods.

2.6.3 Multi-class

For the multi-class classification problem, the output of a DL model will give exactly one label as the output class. The three-class classification task (COVID-19, normal, or pneumonia) has recently become an urgent problem to solve. Nishio et al. [153] achieved 83.68% accuracy using a VGG-16-based model with a combination of data augmentation methods. By starting with a real-time object detection system, named "you only look once," (YOLO), which is based on the Darknet-19 [161] classifier, DarkCovidNet achieved 87.02% accuracy [151]. However, this result could be biased due to the small number of COVID-19 cases

(125) compared to (500) pneumonia cases and (500) normal cases. To compensate for this issue, Mahmud et al. [149] transferred training from a large dataset of normal cases and viral/bacterial pneumonia cases to a small balanced COVID-19 dataset, achieving 90.3% accuracy for their CovXNet model.

Furthermore, COVID-Net [118] leveraged the generative synthesis [162] to determine the optimal design, where the COVID-19 sensitivity and PPV were at or above 80%. Conversely, Oh et al. [163] proposed a patch-based CNN method that may handle the issue of small datasets, as it uses only 11.6 million trainable parameters on COVIDx. Apostolopoulos and Mpesiana [154] achieved 93.48% accuracy by transferring the learning of MobileNet v2 [164]. They concluded that MobileNet v2 was better than VGG-19 [76] for this particular COVID-19 classification task, as it had the fewest instances of False Negatives. Furthermore, Sethy et al. [155] added a Support Vector Machine (SVM) to classify the features obtained from CNN models and achieved 95.33% accuracy. In Chapter 6 (Multi-Class Image Classification), we propose CovidXrayNet, which outperforms benchmark models.

2.6.4 Beyond CXR classification

Classifying multiple image modalities: Table 2.7 summarizes the main characteristics of the multiple-image-modality classifiers. In 2015, the first text/image DL framework with a large-scale PACS was proposed by Shin et al. [65] and used in a national research hospital. This process is explained in more detail in [87]. This system used approximately 780,000 radiology reports and around 216,000 2D images to extract and mine the semantic interactions between them. This framework was capable of matching images with their descriptions automatically using NLP. Latent Dirichlet Allocation (LDA) [165] was applied to obtain the semantic interpretation of diagnostic images, and a CNN was trained to map the images into document categories. The weak supervision method was used to generate interpretations of radiology images, and the strict supervision method was used to detect the absence or presence of several common diseases.

Nevertheless, the clusters in [65] were highly unbalanced. This was because most images were clustered into three groups, as they were derived from text modalities only (approximately 780,000 reports). However, Wang et al. [72] created the LDPO model, which forms clusters from text reports as well as image cues to offer a more visually coherent and balanced method in terms of clustering. As such, LDPO is an iterative system that extracts deep CNN features based on fine-tuned radiologist topic labels and mutual information shared between discovered clusters. When the LDPO model was applied to discovery clusters, visually coherent and highly balanced clusters were observed. However, the looped property is specific to deep CNN classification_clustering methods, as other kinds of classifiers cannot learn satisfactory image characteristics simultaneously.

Generating sentence-level radiology reports: Table 2.8 summarizes the main characteristics of the sentence-level report generators. This area of research was inspired by recent work in generating text descriptions of natural images through intermodal connections between language and visual features [166].

In contrast to the above image classifiers, Shin et al. [26] described the context of a disease in a similar way to a radiology report. They introduced a recurrent neural cascade model to detect and describe the disease location, severity, and the affected organs to offer a better understanding of the disease. This system computed labels based on joint text/image contexts after initial CNN/RNN training using single-object labels from the IU X-Ray dataset [31]. Eventually, it generated image descriptions by training the RNN with the new CNN image embedding (refer to Eq. [(2.3)]), where I is the input image, t is the time step, N is the number of words in the annotation, Y is the output word, S is the correct word and $h_{im:text}$ represents the joint image/text context vector from the first iteration, $iter = 0$.

$$L(I, S) = - \sum_{t=1}^N [P_{RNN_{iter=1}}(Y_t = S_t) | \{CNN_{iter=1}(I) | h_{im:text_{iter=0}}\}] \quad (2.3)$$

TABLE 2.7. The DL Models for Classifying Multiple Image Modalities.

Model	Proposed by	Image Modality	Dataset	Organ	Pathology	Software	CNN Architecture	Base Technique	Task
Deep mining model	Shin, et al. [65] 2015	CT MRI PET Computed radiography Ultrasound	PACS of clinical center [87]	Multiple (e.g., neck, bone, liver, brain and heart)	Multiple (e.g. adenopathy metastasis and sinus diseases)	Caffe [102]	AlexNet [74] VGG-16 [76] VGG-19 [76]	LDA & RNN CNN	Generate semantic labels Map from images to label spaces
LDPO: looped deep pseudo task optimization network	Wang et al. [72] 2016	CT MRI PET Computed radiography Ultrasound	PACS of clinical center [87]	Mutiple (e.g., neck, bone, liver, brain and heart)	Multiple (e.g., adenopathy metastasis and sinus diseases)	Caffe	AlexNet [74]	CNN K-means/RIM NLP	Initialize looped optimization Cluster images Extracts semantically relevant words

TABLE 2.8. The DL Models for Generating Sentence-Level Radiology Reports.

Model	Proposed by	Image Modality	Dataset	Organ	Pathology	Software	CNN Architecture	Base Technique	Task
Recurrent neural cascade model	Shin, et al. [26] 2016	X-Ray	IU X-Ray [31]	Chest	Thorax diseases cardiomegaly, and granuloma)	_	NIN [85] GoogLeNet [77]	CNN LSTM-RNN [91] / GRU-RNN [92]	Classify images Describe disease contexts
Multi-task-loss CNN model	Kisilev, et al. [107] 2016	Mammograph Ultrasound	DDSM Private dataset [34]	Breast	Tumour	Caffe [102]	AlexNet (5 conv. layers) [74]	CNN	Produce ranked ROIs Generate semantic description
Multi-task learning model	Jing, et al. [32] 2017	Multiple	PEIR Gross	21 organ categories (e.g. kidney)	Multiple	_	VGG-19 [76]	CNN MLC	Learn visual features Predict relvent tags

Similarly, the multi-task-loss CNN-based system generated radiologist sentences to describe tumor lesions (shape, margin, and density) in breast images [107]. Essentially, this system was trained using the DDSM dataset and a private dataset of mammography and US images to produce and rank the Rectangular Regions of Interest (ROIs). The highest ROIs were fed into the remaining network layers, which in turn generated semantic descriptions of subsequent ROIs. This system provided automatic lesion detection in breast images alongside semantic descriptions. Jing et al. [32] added a co-attention mechanism to describe abnormal lesions by discovering visual and semantic information.

Generating paragraph-level radiology reports: Table 2.9 summarizes the main characteristics for the paragraph-level report generators. Overall, the purpose of the proposed models is to generate interpretations of radiology images. During training, the input for these models was a collection of images and associated reports. First, researchers proposed models to align disease descriptions to relevant visual regions using multimodal embedding. They then used the outcomes as training data for additional models. This training data allowed the additional models to learn how to generate the image descriptions.

The first work towards generating truly radiology reports with long and diverse topics is a multitask learning model with a co-attention mechanism [32]. It contained a hierarchical LSTM to produce long descriptive paragraphs through capturing long-range semantics. Although this model achieved outstanding results when generating descriptive radiology reports using the IU X-Ray dataset, the produced paragraphs contained repeated sentences due to a lack of contextual coherence in the hierarchical models.

However, Xue et al. [33] generated sentences using the same dataset through an attention input of the image encoding and the first generated sentence. This method maintained coherence in the resultant paragraphs as it used a CNN and LSTM in a recurrent way. As Xue et al. [33] filtered reports without two associated images (frontal and lateral CXRs) and reports with missing information from the IU X-Ray dataset, the training was performed using a small dataset. As a result, the generated text was missing some abnormal descriptions and contained sentences that were different from the ones in the training set.

TABLE 2.9. The DL Models for Generating Paragraph-Level Radiology Reports.

Model	Proposed by	Image Modality	Dataset	Organ	Pathology	Software	CNN Architecture	Base Technique	Task
Multi-task learning model	Jing, et al. [32] 2017	X-Ray	IU X-Ray [31]	Chest	Thorax diseases	_	VGG-19 [76]	CNN Hierarchical LSTM MLC	Learn visual features Generate long paragraphs Predict relevant tags
Multimodal recurrent model with attention	Xue, et al. [33] 2018	X-Ray	IU X-Ray	Chest	Thorax diseases	_	ResNet-152 [78]	CNN Single layer LSTM Bi-LSTM and ID CNN	Extract visual features Sentence decoding Sentence encoding
TieNet: text-image embedding network	Wang, et al. [34] 2018	X-Ray	IU X-Ray	Chest	Thorax diseases	TensorFlow [81] Tensorpack	ResNet-50 [78]	NLP CNN-RNN	Mine disease labels Link words with image regions
HRGR-Agent: hybrid retrieval-generation reinforced agent	Li, et al. [27] 2018	X-Ray	IU X-Ray CX-CHR (Chinese reports) [27]	Chest	Thorax diseases	PyTorch [104]	DensNet [86] VGG19 [76]	LSTM-RNN CNN	Produce reports Extract visual features

Using the same dataset, Wang et al. [34] proposed TieNet, which integrated multi-level attention with a CNN-RNN framework for classification and reporting. The CNN, RNN, and LSTM were based on ResNet-50, the visual spatial attention approach [167], and standard LSTM, respectively. Multiple RNNs may have enhanced TieNet by learning the disease attributes more efficiently, which in turn may have improved the auto-report quality.

Recently, Li et al. [27] introduced the first retrieval model with a generative neural network using RL. It is called the HRGR-Agent. The HRGR-Agent extracts visual features of CXRs from the last convolutional layer of DenseNet or VGG-19 and improves text generation by empowering an RNN with an attention mechanism. Experiments on two medical databases, IU X-Ray and CX-CHR, showed high performance in generating precise text that described rare abnormal findings. The CX-CHR database utilized was a proprietary dataset of Chinese reports and linked images. This made it difficult to compare the HRGR-Agent with other recent state-of-the-art models.

In contrast, Guo et al. [168] used the largest public intensive care unit patient dataset to introduce a framework that learned multiple disease labels from two types of features: medical charts and notes. Instead of considering the correlation between diseases in the same way as existing methods, this approach used disease-specific features. However, the paper only demonstrated an intuitive implementation of the disease-specific feature construction, rather than using multiple clusters for positive and negative instances.

2.7 CXR computer-aided applications

Computer-Aided Detection uses a computer as a tool to generate output that can assist clinicians in making a definitive diagnosis [169]. Unlike automated computer diagnosis, CAD's final diagnosis is not based on computer algorithms alone. Researchers sometimes use Computer-Aided Detection (CADe) and Computer-Aided Diagnosis (CADx) interchangeably. However, CADe minimizes the risk of missing diseases of interest by marking abnormal areas in images, and CADx offers assessment and pathology classification in medical images. There are several CXR applications associated with the CAD, including the applications discussed

in this chapter (i.e., labeling reports in section 2.5, classifying images in section 2.6, and generating reports in section 2.6.4) and the applications that are beyond the scope of this thesis (i.e., segmentation, localization, and image generation).

Segmentation in CXR images refers to segmenting the anatomy to obtain the ROIs for various purposes, such as detecting pulmonary nodules [169]. For example, researchers may aim to segment the lung fields [170], the contours of the lung fields [171], the ribs [172] or the diaphragm. Among the various types of CXR image segmentation, lung field segmentation is essential, as it is related to identifying the ROIs that are linked to lung opacity, consolidation, cavities, and nodules.

Localization in CXR images can be described as the identification of a specific region within the CXR image by a bounding box or a point location [173]. It is an important application to define foreign objects (e.g., catheters) [174], anatomical regions (e.g., ribs) [175], and abnormalities (e.g., nodules) [176] and is easier to accomplish than precise segmentation.

Image generation in CXR images refers to generating new and realistic images to get more interpretable images (e.g., enhancing resolution and removing noise), to obtain additional images for training (e.g., augmenting data), or to increase task performance (e.g., detecting abnormalities) [177]. Since Goodfellow et al. [178] introduced the generative adversarial network in 2014, image generation has become a popular research topic in the medical imaging community [179].

2.8 Evaluation

Evaluating report labeling, image classification, and image captioning models has become increasingly essential due to the rapid introduction of DL approaches to large medical datasets. Both quantitative (machine-based) and qualitative (human-based) evaluations have been employed to compare the benchmark models. Qualitative evaluation is more expensive than quantitative evaluation and is not repeatable. However, it may offer additional valuable measurements for DL outcomes.

2.8.1 Quantitative classification metrics

The most common evaluation metrics for classification tasks are the accuracy, precision, recall, specificity, F1, AUC [180], and Matthews Correlation Coefficient (MCC) [181]. Table 2.10 compares these metrics for the binary classification task in terms of their purposes and algorithms. Note that tp denotes True Positives classifications, fn refers to False Negatives classifications, tn means True Negatives classifications, and fp presents False Positives classifications. Since accuracy depends mostly on the number of samples in each class, CNN-based models seemingly perform well in the imbalanced datasets, such as CXR datasets. This may result in an inaccurate conclusion. Therefore, a combination of multiple evaluation metrics should be the criterion for selecting the best CXR classifier.

TABLE 2.10. Evaluation Metrics (Binary Classification Measures).

Metric	Purpose	Algorithm
Accuracy	Overall classifier effectiveness	$\frac{tp+tn}{tp+fn+fp+tn}$
Precision	Class agreement with the classifier-positive labels	$\frac{tp}{tp+fp}$
Recall (Sensitivity)	Classifier effectiveness in identifying positive labels	$\frac{tp}{tp+fn}$
Specificity	Classifier effectiveness in identifying negative labels	$\frac{tn}{fp+tn}$
F1	Relations between classifier positive labels and data-positive labels	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp} + \beta^2 fn + fp$
AUC	Classifier effectiveness to limit false classification	$\frac{1}{2} \left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp} \right)$
MCC	Indicator of total unbalanced prediction	$\frac{(tp*tn)-(fn*fp)}{\sqrt{(tp+fn)*(tn+fp)*(tp+fp)*(tn+fn)}}$

Accuracy and AUC metrics are not adequate for a highly imbalanced dataset, such as a CXR dataset. The F1 score, however, combines the strengths of recall (i.e., the ratio of true positive predictions to positive samples) and precision (i.e., the ratio of true positive predictions to the sum of all positive predictions, true and false). Hence, the F1 score can fairly compare

benchmarks regardless of CXR dataset imbalance, where most cases are negative (i.e., healthy patients).

For multi-class classification, Eqs. (2.4), (2.5), (2.6) and (2.7) explain the accuracy, macro average precision, macro average recall, and macro F1 score, respectively, for generic class k . Note that TP refers to True Positives classifications, FN denotes False Negatives classifications, TN presents True Negatives classifications, and FP means False Positives classifications. In the macro approach, all classes are considered basic elements of the calculation [182] (i.e., each class has the same weight in the average regardless of its size).

$$Accuracy = \frac{\sum_{k=1}^K \frac{TN_k + TP_k}{TN_k + TP_k + FN_k + FP_k}}{K} \quad (2.4)$$

$$Precision_{Macro} = \frac{\sum_{k=1}^K \frac{TP_k}{TP_k + FP_k}}{K} \quad (2.5)$$

$$Recall_{Macro} = \frac{\sum_{k=1}^K \frac{TP_k}{TP_k + FN_k}}{K} \quad (2.6)$$

$$F1_{Macro} = 2 \times \frac{Precision_{Macro} \times Recall_{Macro}}{Precision_{Macro}^{-1} + Recall_{Macro}^{-1}} \quad (2.7)$$

Furthermore, the AUC [180] for multi-class classification is defined in Eq. (2.8), where $AUC(c_i)$ is the area under the class reference receiver operating characteristic ROC curve for the positive class, c_i . This implementation of the AUC score is simple and fast but it is sensitive to class distributions and error costs. The MCC, however, is a good indicator of total unbalanced prediction models, as defined in Eq. (2.9), where c represents all correctly predicted cases, s represents all cases, p_k is the number of instances that class " k " was predicted to be, and t_k is the number of instances when class " k " truly occurred.

$$AUC_{total} = \sum_{c_i \in C} AUC(c_i) \times p(c_i) \quad (2.8)$$

$$MCC = \frac{c \times s - \sum_k^K P_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) (s^2 - \sum_k^K t_k^2)}} \quad (2.9)$$

2.8.2 Quantitative captioning metrics

The most common evaluation metrics for image captioning and machine learning are Bilingual Evaluation Understudy (BLEU) [183], Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [184], METEOR [185], Consensus-Based Image Description Evaluation (CIDEr) [186], and Semantic Propositional Image Caption Evaluation (SPICE) [[187]. Table 2.11 compares these metrics using their original purposes, main ideas, strengths, and weaknesses.

These evaluation metrics are employed by researchers to compare their proposed models of generating radiology reports against the benchmarks. They automatically calculate an accuracy score for a new model by observing the similarities and differences between the generated captions and the radiologist’s written descriptions from empirical observation. Increased performance is indicated through higher scores in BLEU, ROUGE, METEOR, CIDEr, and SPICE. The MS COCO evaluation kit³⁶ offers the implementation script for these evaluation metrics in terms of caption generation.

The BLEU-n metrics are precision metrics for machine translation that are computed by multiplying n-gram precision scores by a penalty for short sentences. They have been employed to measure the similarity between a pair of sentences. A superior version of BLEU was proposed by Lin et al. [188]. However, BLEU suffers from a low performance in explicit word matching.

The ROUGE metric is a recall metric for summarization systems that matches intersecting n-grams, word sequences, and word pairs; ROUGE-L is a version of ROUGE that calculates the longest common sub sequences between two sentences.

The METEOR metric is a recall metric for machine translation that utilizes synonyms, paraphrase matching, precision, and unigram recall to obtain harmonic overlapping between

³⁶<https://github.com/tylin/coco-caption>.

sentences. It overcomes BLEU's weaknesses in failing to locate semantic similarity by applying synonym matching based on WordNet. Nonetheless, observing synonyms alone may not be adequate to capture semantic similarities.

The CIDEr metric is an evaluation metric for image captioning that calculates cosine similarity between candidate image c_i annotation and the associated sentences produced by humans. It works via purely linguistic means, but its evaluations are ineffective as it sometimes provides large weighting for insignificant sentence details.

The SPICE metric is a recent evaluation metric for image captioning that uses scene-graph tuples to parse a sentence into semantic tokens including object classes, relation types, and attribute types. Thus, the quality of the parsing determines CIDEr's performance. In some cases, this may result in failure, as illustrated by an example in Kilickaya et al.'s work [189]. Similar to METEOR, SPICE utilizes WordNet synonym matching for tuple matching.

The different design choices of evaluation metrics, such as n-grams and scene-graphs, result in metrics that have different strengths and weaknesses. For example, BLEU, ROUGE, and CIDEr use only exact n-gram matches, but METEOR adds synonyms and phrases. Although BLEU is based on precision, METEOR and ROUGE are recall-based metrics. As a result, Kilickaya et al. [189] suggested that existing evaluation metrics should complement each other in measuring the quality, accuracy, and robustness of the generated annotations. Table 2.12 compares the results of the reports generated by the models in Table 2.9 through quantitative evaluation metrics.

However, the original purpose of these common metrics was not to evaluate generated radiology reports. Therefore, some researchers have designed complementary metrics. For instance, a metric called keywords accuracy calculates accuracy by dividing the number of correctly generated words by the number of ground truth words from the Medical Text Indexer annotations [33].

TABLE 2.11. Evaluation Metrics (Image Caption Measures).

Metric	Purpose	Algorithm	Strengths	Weaknesses
BLEU [183] 2002	Machine translation	N_{gram} precision	Correlates with human judgments	Lack of explicit word matching
ROUGE [184] 2004	Document summarization	N_{gram} recall	Favors long sentences	Works only in single-document summarization
METEOR [185] 2005	Machine translation	N_{gram} with synonym matching	Benefits from synonyms and phrase matching	Lack of semantic similarity capturing
CIDEr [186] 2015	Image captioning	N_{gram} with corpus reweighting	Works via linguistics means	May weight irrelevant sentences'details
SPICE [187] 2016	Image captioning	$f_{objects} * f_{attributes} * f_{relations}$	Can match nouns / objects between captions	Reliant on the performance of parsing

TABLE 2.12. Quantitative Evaluation of Generated Radiology Reports based on DL Models.

Model	Database	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGH	ROUGH_L	CIDER
Sentence-level									
Recurrent neural cascade model	LSTM [26]	IU X-Ray [31]	0.793	0.091	0.000	0.000	–	–	–
Recurrent neural cascade model	GRU [26]	IU X-Ray	0.785	0.144	0.047	0.000	–	–	–
Multi-task learning model [32]		PEIR	0.300	0.218	0.165	0.113	0.149	0.279	–
Paragraph-level									
Multi-task learning model [32]		IU X-Ray	0.517	0.386	0.306	0.247	0.217	0.447	–
Multimodal recurrent model with attention [33]		IU X-Ray	0.464	0.358	0.270	0.195	0.274	0.366	–
TieNet [34]		IU X-Ray	0.286	0.160	0.104	0.074	0.108	–	0.226
HRGR-Agent [27]		IU X-Ray	0.438	0.298	0.151	–	0.322	–	–
		CX-CHR	0.673	0.587	0.530	0.486	–	0.612	–

2.8.3 Qualitative measures

Qualitative evaluation involves comparing ground-truth reports with model-generated reports using content coverage, length, medical term accuracy, and text fluency. For example, Li et al. [27] utilized Amazon Mechanical Turk to conduct surveys. Here, participants chose the generated report that best matched the ground truth report. Jing et al. [32] manually compared the generated paragraphs from their co-attention model with the ground truth to establish which models captured normality and abnormality most efficiently.

2.9 Discussion and future directions

Deep learning algorithms have the potential to be used in all fields of medicine and could significantly alter the way medicine is practiced. Future DL research should utilize the wealth of medical images and relevant diagnostic reports that were recently released to automatically interpret CXRs. Recent attention has focused on classifying CXR images due to the value of this task to the healthcare system.

In a radiology database, the data is unbalanced because abnormal cases are rarer than normal cases. For example, the healthy cases in the IU X-Ray dataset consist of 2,696 images (37%) compared to the 840 images (12%) that represent common diseases and 655 images (9%) that show less common diseases. Shin et al. [26] attempted to address this issue by training CNNs with different regularization methods including batch normalization and data dropout. In addition, it is challenging to automate labels for medical images as radiologist reports often include ambiguous words. This includes disease predictions rather than clear indications as to whether a disease is present [87]. It should be noted that it is difficult to compare various models because researchers conduct their experiments using diverse and sometimes private datasets. For example, researchers conduct experiments using different database subsets. This makes it difficult to compare the performance of their proposed approaches.

Researchers consider DL as a black box that takes an input, such as a medical image, and generates an output to state a conclusion (e.g., “There is a 0.8 probability of melanoma”) without

clear explanations [23][190]. This is unacceptable in the medical domain, as radiologists need to provide findings as well as underlying justifications. For instance, researchers may attempt to provide the rationale behind a radiologist's description using their proposed models. Considerably more research will need to be conducted to offer reasonable explanations for DL model outcomes.

Most research uses CNNs to apply text-image mining in medical imaging. As such, CNNs have the widest variety of architecture, including AlexNet, VGG-16, GoogLeNet, and ResNet. In the last three years, end-to-end trained CNN use has become the preferred approach for medical imaging interpretation. As such, this could be considered standard practice for mining medical images. In addition, it is likely that the volume of research in leveraging radiology reports for CNN training will increase in the near future.

Deep learning has several limitations that should be addressed to improve the task of CXR interpretation. A reliable AI system may require tens of millions of accurately labeled images which are not yet readily available [23]. Furthermore, these samples should be structured without scattered or noisy information to facilitate the learning process for DL models. To date, there are few medical datasets that are large and accessible enough to train multimodal deep CNNs. Improving the quantity and quality of radiology data remains an ongoing task.

Creating multipurpose reporting systems for radiologists that can detect several diseases is another current challenge. Medical findings often correlate with certain body parts such as metastatic spread to the liver and lymph nodes. Despite the promising results of interpreting CXR image interpretations, several questions must be addressed. For example, what are the clinically related image annotations to be defined? How should the large volume of radiology images required for DL techniques be labeled? To what extent is the deep CNN framework generalizable for radiology images? Future work should explore valuable semantic diagnostic information and map the many well-written radiologist reports and relevant images available.

Beyond medical image classifications and single-sentence-based descriptions, generating radiology-coherent paragraphs has recently attracted researchers' interest. This presents a more practical and challenging application that can bridge visual medical features with

radiologist interpretations. Notably, CNNs and RNNs have quickly become popular choices for mining radiology images and text, respectively. The main challenge now lies in how to obtain ImageNet-level semantic labels on a large collection of medical images.

2.10 Conclusion

This chapter presented a comprehensive literature survey of DL in CXR interpretations to place and justify our contributions in the following chapters. Deep learning in CXR interpretations is crucial, as DL techniques can quickly and accurately provide additional diagnostic criteria by reporting observable data from the images and text.

The rest of this thesis is structured as follows: In Chapter 3, we introduce our CXR text-labeling tool. Then, we describe our CXR classifiers: the binary classifier in Chapter 4, the multi-label classifier in Chapter 5, and the multi-class classifier in Chapter 6. The thesis concludes with Chapter 7.

Report Labeling¹

The path to the development of Chest X-ray (CXR) interpretation models based on Deep Learning (DL) starts with extracting multiple image annotations from radiology reports. This chapter proposes CXRlabeler, a labeler that extracts multiple observations from CXR reports. It fine-tunes a pre-trained Language Model (LM) to the corpus of radiology impressions and then uses the LM encoder with a new head to classify CXR reports.

3.1 Introduction

There is an urgent need to automatically label large CXR datasets using radiology reports to facilitate the training of deep neural networks. However, this task creates a significant bottleneck as it requires considerable medical skills and time to output high-quality information. Fortunately, Natural Language Processing (NLP) methods offer the ability to annotate free-text reports automatically. Researchers have implemented several rule-based NLP and expert-defined labeling systems [13][121] to extract labels from CXR reports to be used as structured labels for CXR images. Even though these systems have been helpful to researchers, they have several problems, such as low accuracy and a slow run time [130].

Currently, there are three public CXR datasets that release complete radiology reports, i.e., Indiana University Chest X-Ray (IU X-Ray) [31], PATHology Detection in Chest radiographs (PadChest) [115], and Medical Information Mart for Intensive Care Chest X-ray (MIMIC-CXR) [113] datasets, as described in section 2.4. Figure 3.1 shows an example

¹The content in this chapter has been published in the International Conference on Artificial Neural Networks (ICANN), "Labeling Chest X-Ray Reports Using Deep Learning", Monshi, M., Poon, J., Chung, V., Monshi, F. (2021).

Report	Labels																												
<p>PORTABLE AP CHEST FILM, — AT 11:18</p> <p>CLINICAL INDICATION: —year-old status post CABG, status post chest tube removal, question pneumothorax. A portable AP upright chest film, — at 11:18 is submitted.</p> <p>IMPRESSION: 1. Interval removal of the left chest tube. No evidence of pneumothorax. Right internal jugular central line has its tip in the distal SVC near the cavoatrial junction, unchanged. Status post median sternotomy for CABG with stable postoperative cardiac and mediastinal contours. There is elevation of the left hemidiaphragm with some adjacent streaky opacities, suggestive of atelectasis. Blunting of the left costophrenic angle likely reflects a small effusion. There is also possibly a tiny right pleural effusion. No evidence of pulmonary edema.</p>	<table> <tr><td>Atelectasis</td><td>1.0</td></tr> <tr><td>Cardiomegaly</td><td>NaN</td></tr> <tr><td>Consolidation</td><td>NaN</td></tr> <tr><td>Edema</td><td>0.0</td></tr> <tr><td>Enlarged Cardio.</td><td>-1.0</td></tr> <tr><td>Fracture</td><td>NaN</td></tr> <tr><td>Lung Lesion</td><td>NaN</td></tr> <tr><td>Lung Opacity</td><td>1.0</td></tr> <tr><td>No Finding</td><td>NaN</td></tr> <tr><td>Pleural Effusion</td><td>1.0</td></tr> <tr><td>Pleural Other</td><td>NaN</td></tr> <tr><td>Pneumonia</td><td>NaN</td></tr> <tr><td>Pneumothorax</td><td>0.0</td></tr> <tr><td>Support Devices</td><td>1.0</td></tr> </table>	Atelectasis	1.0	Cardiomegaly	NaN	Consolidation	NaN	Edema	0.0	Enlarged Cardio.	-1.0	Fracture	NaN	Lung Lesion	NaN	Lung Opacity	1.0	No Finding	NaN	Pleural Effusion	1.0	Pleural Other	NaN	Pneumonia	NaN	Pneumothorax	0.0	Support Devices	1.0
Atelectasis	1.0																												
Cardiomegaly	NaN																												
Consolidation	NaN																												
Edema	0.0																												
Enlarged Cardio.	-1.0																												
Fracture	NaN																												
Lung Lesion	NaN																												
Lung Opacity	1.0																												
No Finding	NaN																												
Pleural Effusion	1.0																												
Pleural Other	NaN																												
Pneumonia	NaN																												
Pneumothorax	0.0																												
Support Devices	1.0																												

FIGURE 3.1. Example of a Labeled Report from the MIMIC-CXR Dataset. Each label contains one of four values, 1.0, -1.0, 0.0, or *NaN*, which indicate positive, negative, uncertain, or missing observations, respectively [132].

of a labeled report from the MIMIC-CXR dataset. Each label contains one of four values, i.e., 1.0, -1.0, 0.0, or *NaN*, which indicate positive, negative, uncertain, or missing observations, respectively.

For multilabel classification in the radiology domain, the task is to predict the probability of multiple target medical observations linked to a report. This type of classification is a common task in many real-life domains, such as in extracting probable diseases based on observed clinical symptoms and automatic document tagging [191]. Chest X-ray reports are typically semi-structured, where the impression section summarizes the most relevant findings, describing multiple observations and important features of the CXR [28].

Several NLP systems have been proposed for extracting medical labels from CXR reports. These are based on feature engineering, such as NegBio [121] and CheXpert [13], or DL algorithms, such as Recurrent Neural Network with Attention (RNN-ATT) [115], CheXpert++

[130], and CheXbert [131]. We described these labelers in the previous chapter (i.e., section 2.5).

3.1.1 Contributions

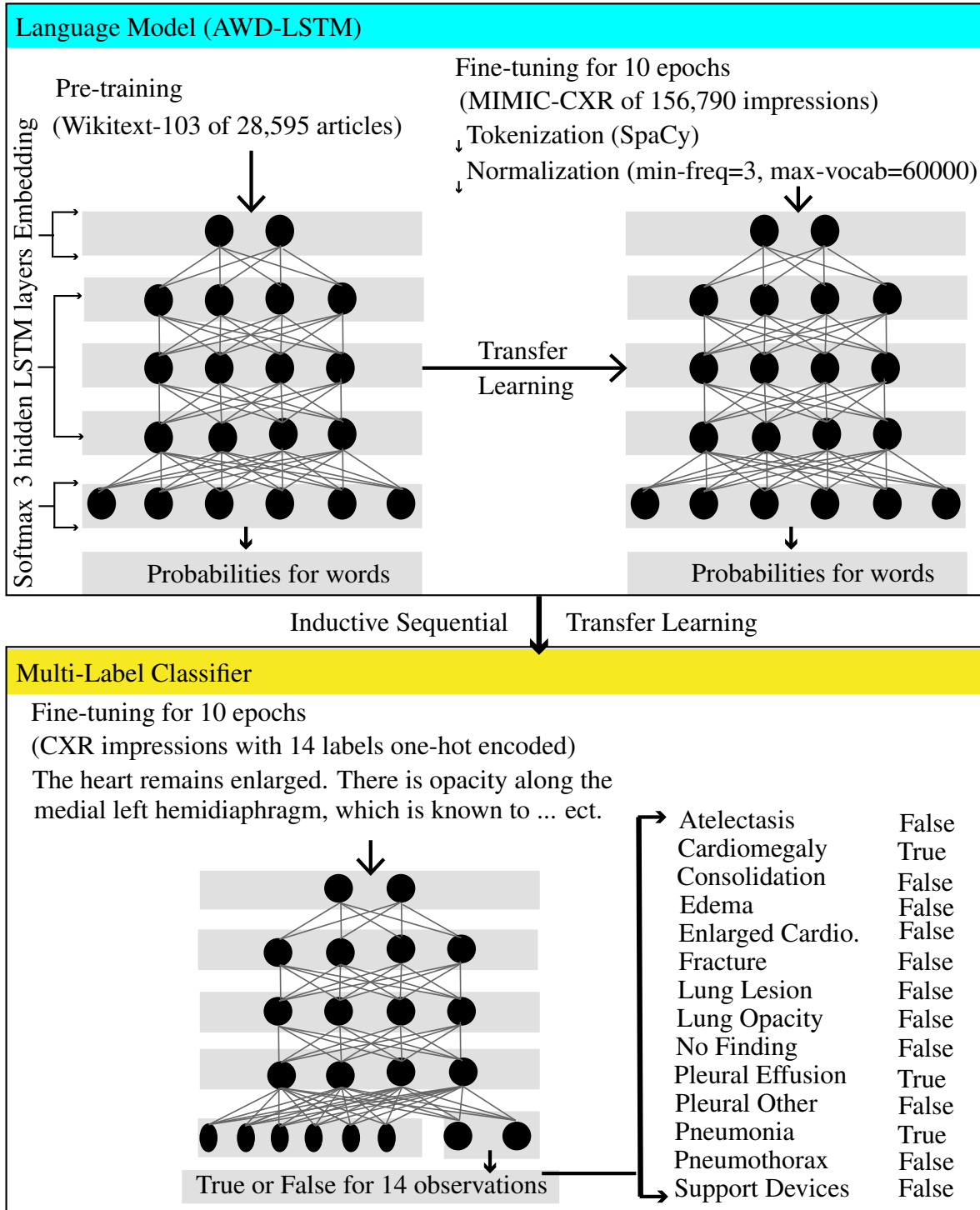
Differing from rule-based methods, which cannot handle the extensive linguistic ambiguity in radiology reports, including misspellings and broken grammar, the work in this chapter is related to DL approaches for labeling CXR reports. Deep learning models have been shown to be beneficial in the training of high-quality models even in the absence of trustworthy training labels [192].

This chapter proposes a novel model named CXRlabeler. It is a DL labeler that inputs raw radiology text and extracts multiple positive/negative CXR observations as its output. For labeling these reports, it utilizes the encoder learned from fine-tuning an LM on CXR reports. For highly accurate automated CXR report labeling, our proposed method takes advantage of both LM fine-tuning and classifier fine-tuning.

First, our CXRlabeler uses a popular pre-trained LM, named the Averaged Stochastic Gradient Descent Weight-Dropped Long Short-Term Memory (AWD-LSTM) model [193]. Second, CXRlabeler fine-tunes the AWD-LSTM using a corpus of CXR reports. Lastly, CXRlabeler uses the AWD-LSTM body (i.e., the encoder) with a new head to classify the impression sections of the CXR reports.

3.2 Proposed CXRlabeler model

As the input, CXRlabeler takes a radiology text with pre-tagged mentions of CXR observations and checks whether a particular observation is positive or negative. This study used the pre-trained LM/AWD-LSTM network. Figure 3.2 represents the overall pipeline of CXRlabeler. The detailed steps are explained in the following subsections.



3.2.1 Data preparation

From the MIMIC-CXR dataset (refer to section 2.4.4), this study extracted 152,855 unique impressions for training and 3,935 for testing (merging the official validation and test split). A binary mapping approach was followed, called the U-Zeros model [13], where uncertain and missing values are mapped to negative instances. Table 3.1 records the frequency of the 14 labels in the preprocessed MIMIC-CXR dataset.

TABLE 3.1. Frequency of the 14 Labels in the Preprocessed MIMIC-CXR Dataset. The study extracted 156,790 unique impressions, of which 152,855 were used for training and 3,935 for testing. It reported the number of positive and negative cases for each label, with missing and uncertain labels considered negative labels.

Label	Positive			Negative		
	Train	Test	%	Train	Test	%
Atelectasis	42,052	1,038	27	110,803	2,897	73
Cardiomegaly	40,970	1,160	27	111,885	2,775	73
Consolidation	9,962	293	7	142,893	3,642	93
Edema	24,035	842	16	128,820	3,093	84
Enlarged Cardio.	6,589	199	4	146,266	3,736	96
Fracture	4,040	99	3	148,815	3,836	97
Lung Lesion	5,843	183	4	147,012	3,752	96
Lung Opacity	47,757	1,368	31	105,098	2,567	69
No Finding	24,487	472	16	128,368	3,463	84
Pleural Effusion	50,035	1,497	33	102,820	2,438	67
Pleural Other	1,851	71	1	151,004	3,864	99
Pneumonia	14,665	433	10	138,190	3,502	90
Pneumothorax	9,644	178	6	143,211	3,757	94
Support Devices	61,768	1,634	40	91,087	2,301	60

From the PadChest dataset (refer to section 2.4.5), nine labels were extracted that matched those of the MIMIC-CXR dataset (atelectasis, cardiomegaly, consolidation, edema, fracture, pleural effusion, pneumonia, and pneumothorax). As a result, the study ended up with 121,808 unique Spanish reports for training and 39,053 for testing, drawn from the PadChest reports manually labeled by qualified radiologists. Table 3.2 reports the frequency of the nine labels in the preprocessed PadChest dataset.

After performing similar data preparation on the IU X-Ray dataset (refer to section 2.4.1), we ended up with 1,776 unique reports. This was a relatively small dataset and was attributed to the many missing sections in the IU X-Ray dataset, which was derived from two hospitals after a full anonymization process.

TABLE 3.2. The Frequency of Nine English Labels in the Preprocessed PadChest Dataset. The study extracted 29,365 unique Spanish reports, of which 22,275 were for training and 7,090 for testing. The testing split was manually labeled [115].

Label	Positive			Negative		
	Train	Test	%	Train	Test	%
Atelectasis	4,401	1,471	20	17,874	5,619	80
Cardiomegaly	5,898	2,300	28	16,377	4,790	72
Consolidation	1,370	232	5	20,905	6,858	95
Edema	1,169	71	4	21,106	7,019	96
Fracture	2,101	751	10	20,174	6,339	90
No Finding	3,042	1,303	15	19,233	5,787	85
Pleural Effusion	5,520	1,193	23	16,755	5,897	77
Pneumonia	3,854	1,084	17	18,421	6,006	83
Pneumothorax	275	67	1	22,000	7,023	99

3.2.2 Language model

Before a multi-label classifier was built, the LM was pre-trained on Wikitext-103 [194], containing 28,595 preprocessed Wikipedia articles, and 103 million words. Then, the LM was fine-tuned to the radiology corpus to introduce the LM to the medical language, medical terms and informative sentence structures. This recently introduced fine-tuning approach has been found to enhance the classifier prediction significantly [195]. For the PadChest dataset, the LM was pre-trained on Spanish Wikipedia with a 15,000-word vocabulary.

The LM was fine-tuned on the entire CXR radiology corpus regardless of the data split following the transfer learning and freezing/unfreezing layer protocol [195]. For this self-supervised learning, all impression sections in the MIMIC-CXR dataset were fed to the LM without labels, as it was able to get labels from the data automatically. The resulting LM

could predict the next word in a radiology report based on previous words with an accuracy of 62.92% for MIMIC-CXR and 49.99% for PadChest.

3.2.3 Multi-label classifier

In multilabel classification, the task is to learn a function, $h : X \rightarrow 2^Y$, which assigns a subset of related medical observations from a finite set of Q predefined labels, $Y = \{y_1, y_2, \dots, y_Q\}$, to each report from an instance space, $X = \{x_1, \dots, x_N\}$, $X \subseteq R^n$. In this study, first, the learned encoder was loaded from the LM, which was then used for vocabulary and text. Second, all of the pre-trained layers were frozen, and the last layer was trained for one epoch. Finally, all layers were unfrozen and trained for 10 epochs while adjusting the learning rates using the discriminative learning rates protocol [195]. This protocol enabled us to optimize the neural net efficiently by training it with different learning rates for various layers.

This approach was based on inductive sequential transfer learning, where the source and the target task are not the same and the source data's general knowledge is transferred to a single task [196]. Howard et al. [195] concluded that this method of transfer learning has led to the most significant improvements in text classification benchmarks by decreasing errors by an incredible 18%_24% on several datasets.

3.3 Experiment

For the DL training, this study used the PyTorch software [103], the Fastai v2 library [105], and an n1-highmem-8 (8 vCPUs, 52 GB memory) machine with a single NVIDIA Tesla V100 Graphics Processing Unit (GPU). Fastai outperforms existing DL libraries in handling the multi-label text classification tasks [197]. To avoid overfitting, the training was continued while the validation loss was lower than the training loss. From the Learning Rate (LR) finder graph, the study selected the middle point of the most significant downward slope as the LR.

Our proposed CXRlabeler is available in Appendix (B), and more details about the implementation of this model can be found on <https://github.com/MaramMonshi/CXRlabeler>.

3.4 Results and discussion

The accuracy, macro Area Under the Receiver Operating Characteristic Curve (AUC), macro precision, macro recall, and macro F1 score were used to evaluate CXRlabeler. We explained these metrics in the previous chapter (i.e., section 2.8.1). Each matrix gave valuable insight into CXRlabeler’s performance. However, the accuracy and AUC metrics are not adequate for a highly imbalanced dataset, as shown in Table 3.1 and Table 3.2. Therefore, the thresholds for accuracy were set to above 0.8 to build a robust model. The F1 score, however, combines the strengths of recall and precision. As a result, it could fairly compare CXRlabeler with the benchmarks despite the CXR dataset imbalance issue, where most cases were healthy patients.

Table 3.3 compares CXRlabeler with other models on the multi-label classification task. Previous labelers used input sentences from a CXR report, and the output for each finding was one of the following classes: positive, negative, uncertain, or blank. For CXRlabeler, the outputs were one of the two following classes: positive or negative. CXRlabeler achieved an F1 of 96.17%, significantly higher than the baseline performance of RNN-ATT (60.10%), CheXpert++ (79.10%), CheXbert (79.80%), NegBio (94.40%), and CheXpert (94.80%).

Moreover, transfer learning a fine-tuned LM increased the classifier F1 score by 12.53%, as recorded in Table 3.3. To pre-train CXRlabeler, we applied AWD-LSTM to an unlabeled corpus of 156,790 CXR impressions from the MIMIC-CXR dataset. This pre-training stage on CXR reports is especially important, as spelling and abbreviations vary greatly between CXR reports and Wikitext-103 [198]. According to Drozdov et al. [199], including more reports in the pre-training corpus may improve performance even more in the CXR report labeling task.

TABLE 3.3. Comparing CXRlabeler with the Benchmarks in Labeling CXR Reports. CXRlabeler classifies each label as positive or negative.

Model	Dataset	Accuracy	AUC	Precision	Recall	F1
NegBio [121]	OpenI	–	–	89.80	85.70	87.30
NegBio [121]	ChestX-ray14	–	–	94.40	95.70	94.40
CheXpert [13]	CheXpert	–	–	–	–	94.80
CheXpert++ [130]	CheXpert	–	–	–	–	79.10
RNN-ATT [115]	PadChest	86.40	–	–	–	60.10
CXRlabeler (without LM)	PadChest	95.26	92.94	65.53	58.80	60.06
CXRlabeler (with LM)	PadChest	99.16	98.95	96.55	89.01	92.08
CheXbert [131]	MIMIC-CXR	–	–	–	–	79.80
CheXpert [13]	MIMIC-CXR	–	–	79.30	91.16	82.54
CXRlabeler (without LM)	MIMIC-CXR	97.58	98.10	87.62	81.24	83.64
CXRlabeler (with LM)	MIMIC-CXR	99.23	99.80	95.92	96.45	96.17

Table 3.4 illustrates some examples of the labels extracted from radiology reports using the automated labeling system of CXRlabeler. In most cases, the labeler extracted all observations correctly. Benchmark NLP labelers NegBio and CheXpert were built and validated using the ChestX-ray14 and CheXpert datasets, respectively. These datasets released images and associated labels without the associated radiology reports. Since this study did not have access to these reports, CXRlabeler was evaluated using the MIMIC-CXR, and the PadChest datasets. In addition, CXRlabeler achieved a very low F1 score of 37.73% on the IU X-Ray because of the dataset size limitation described earlier.

The CXR reports in the MIMIC-CXR dataset describe important CXR findings with respect to anatomical locations using spatial prepositions, as presented in Table 3.4. For example, the phrase “mild streaky opacities are present in the left lung base” states the CXR findings (i.e., opacities) in reference to the anatomy (i.e., the left lung base) through a spatial preposition (i.e., in). However, spatial language understanding remains little studied in the radiology domain due to the complexity of the language used to define spatial relations [26][200]. Recently, Datta et al. [201] employed the spatial role labeling method [202] to extract the spatial information related to CXR findings, providing a preliminary step to understanding the textual spatial semantics in CXR reports.

TABLE 3.4. Examples of the Labels Predicted by CXRlabeler and the Target Labels.

Ex.	Input	Target	Predicted	Result
1	Large right and moderate left pleural effusions and severe bibasilar atelectasis are unchanged. Cardiac silhouette is obscured. No pneumothorax. Pulmonary edema is mild.	Atelectasis, Edema, Pleural Effusion	Atelectasis, Edema, Pleural Effusion	Correct
2	1. Slowly growing peripheral right upper lobe lung nodule is concerning for primary lung adenocarcinoma . 2. Low lung volumes limit assessment of the lung bases for pneumonia .	Lung Lesion, Pneumonia	Lung Lesion, Pneumonia	Correct
3	xxmaj the pre-existing very diffuse bilateral interstitial opacities , likely reflecting interstitial lung edema , are unchanged. xxmaj this is supported by the presence of pleural effusion , evident on the lateral image . xxmaj however , in the right lung , subtle calcified granulomas are present . xxmaj the patient also shows several calcified mediastinal lymph nodes .	Cardiomegaly, Edema, Lung Opacity, Pleural Effusion, Pneumonia, Support Devices	Cardiomegaly, Edema, Lung Opacity, Pleural Effusion, Pneumonia	Wrong (missed support devices)

Advanced CXR report labelers lead to higher-performing models that interpret CXR images. Jain et al. [203] demonstrated how robust CXR labelers (particularly VisualCheXbert [204]) can assist in further increasing the accuracy of CXR classification models. In the future, DL models will be trained on labels generated by CXRlabeler and recent automatic CXR report labelers.

This study followed the binary assignment of 0s (negative) and 1s (positive), where the uncertainty label did not add information to the classifier and, hence, may have degraded the decision-making performance. Irvin et al. [13] suggested treating the uncertain label, u , as a separate class to better disambiguate the uncertain cases, where the probability output of the three classes is $p_0, p_1, p_u \in [0, 1]$, $p_0 + p_1 + p_u = 1$. In the future, the scope of this method will be extended to detect uncertainty, which is critical knowledge in the medical domain.

3.5 Summary and conclusion

One of the primary challenges in the development of CXR interpretation models is the lack of large datasets with multi-label image annotations extracted from CXR reports. This chapter proposed CXRlabeler, which can simultaneously extract multiple observations from free-text radiology reports as positive or negative and is abbreviated as CXRlabeler. It fine-tuned a pre-trained LM/AWD-LSTM, to the corpus of CXR radiology impressions and then used it as the base of the multilabel classifier. Experimentation demonstrated that the LM fine-tuning increased the classifier F1 score by 12.53%. Overall, CXRlabeler achieved a 96.17% F1 score on the MIMIC-CXR dataset. To further test the generalization of CXRlabeler, it was tested on the PadChest dataset. This testing showed that the CXRlabeler approach was helpful in a different language environment.

Binary Image Classification¹

In the previous chapter, we looked at the process of labeling Chest X-ray (CXR) images by extracting labels from CXR reports. In this chapter, we look at interpreting CXRs using large-scale labeled images. We propose a stage-wise model, named MultiViewModel, that is founded on a Residual Network (ResNet)-based deep Convolutional Neural Network (CNN) architecture [78] to detect the presence and absence of thorax diseases. This novel binary classifier incorporates various recent techniques, such as transfer learning, fine-tuning, fit-one-cycle functions [205], and discriminative Learning Rate (LR) [195].

4.1 Introduction

Currently, CXR analysis depends on the availability of professional radiologists. In some regions, access to such radiologists is limited [13]. Additionally, clinicians in emergency departments and intensive care units need fast and accurate interpretations of medical images [106]. An automated and precise binary classifier that can flag potentially life-threatening diseases could allow care providers to handle emergency cases efficiently.

However, interpreting CXRs to detect thoracic diseases is still a challenging job. This is due to the highly diverse appearance of lesion areas on CXRs. Unlike the traditional Computer-Aided Detection (CAD) systems that interpret medical images automatically to offer an objective diagnosis that assists radiologists [15], Deep Learning (DL) is able to learn useful features that are beyond the limits of radiology detection [16]. Researchers have shown

¹The content in this chapter has been published in International Conference on Neural Information Processing (ICONIP), "Labeling Chest X-Ray Reports Using Deep Learning", Monshi, M., Poon, J., Chung, V. (2019).

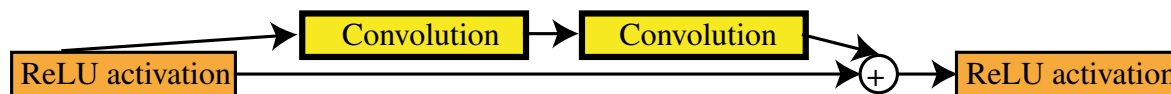


FIGURE 4.1. A Basic Residual Block [146].

a significant performance boost using their DL-based models over the conventional CAD systems [17][18].

As a result, several DL models that classify thorax diseases have been proposed, as explained in Chapter 2 (Literature Review), section 2.6.1. Most of these models classify thorax diseases from frontal CXRs using the ChestX-ray14 dataset [110][108][144][145][147]. However, recent studies have pointed out that using lateral views enhances the performance for certain prediction tasks, such as those for pleural effusion [106][116][157]. Therefore, we used multiple CXR views to build our binary classifier model.

A Convolutional Neural Network (CNN) which is a supervised DL model, is the most commonly used network for thoracic disease classification. Convolutional neural networks also have the largest variety of architectures, as discussed in Chapter 2 (Literature Review), section 2.3.2.1. Moreover, ResNet is the CNN architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015 with a 3.6% top-five error rate [78]. It enabled automated image classification to beat human brains within 5% error for the first time. It is a feed-forward network that contains several basic residual blocks (refer to Fig. 4.1) to handle vanishing gradients [83] and the degradation issue. We built our binary classifier based on ResNet.

4.1.1 Contributions

In this chapter, we present a supervised DL model using CNN to detect 12 thoracic diseases by reading a given CXR. Here, ResNet-50 was the backbone network for our model because it has clearly shown outstanding performance in computer vision.

Consistent with recently proposed CNN models on automated CXR classification [106][108]-[66], we focused on training CNN models to detect 12 common thoracic diseases, namely

enlarged cardiomeastinum, cardiomegaly, airspace opacity, lung lesion, edema, consolidation, pneumonia, atelectasis, pneumothorax, pleural effusion, pleural other and fracture (Fig. 4.2).

Unlike past works, we propose MultiViewModel, a novel stage-wise training approach, to observe the model’s performance, reduce the training time, and increase accuracy. We trained our model on multiple CXR views, including Posteroanterior (PA), Anteroposterior (AP), and lateral views. Furthermore, we adopted a combination of recent DL techniques for CXR classification, including transfer learning, fine-tuning, fit-one-cycle functions, and discriminative LR.

4.2 Proposed MultiViewModel

4.2.1 Data preparation

We organized a subset of 10% of the Medical Information Mart for Intensive Care Chest X-ray (MIMIC-CXR) v1.0.0 dataset (refer to section 2.4.4) into training and validation sets that contained 33,195 and 3,688 images, respectively. The validation set was selected at random. During training, the uncertain and unknown labels were ignored. Table 4.1 shows the positive and negative cases for each observation.

Prior to the model training, we employed several augmentation strategies (refer to Table 4.2), as data augmentation is a critical step of deep CNN training in medical imaging [206]. We cropped each X-ray in both the training and validation sets to 224 by 224 pixels to reduce the training time while maintaining the model’s robust performance. For example, training the model to diagnose cardiomegaly using images of 299×299 pixels would have increased the training time without improving the Area Under the Receiver Operating Characteristic Curve (AUC) per epoch (refer to Table 4.3). We performed a horizontal flip only for each image in the training set, since vertical flips often do not reflect CXRs (i.e., an upside-down CXR may not improve the training). The maximum lighting of the image was set to 0.3 with

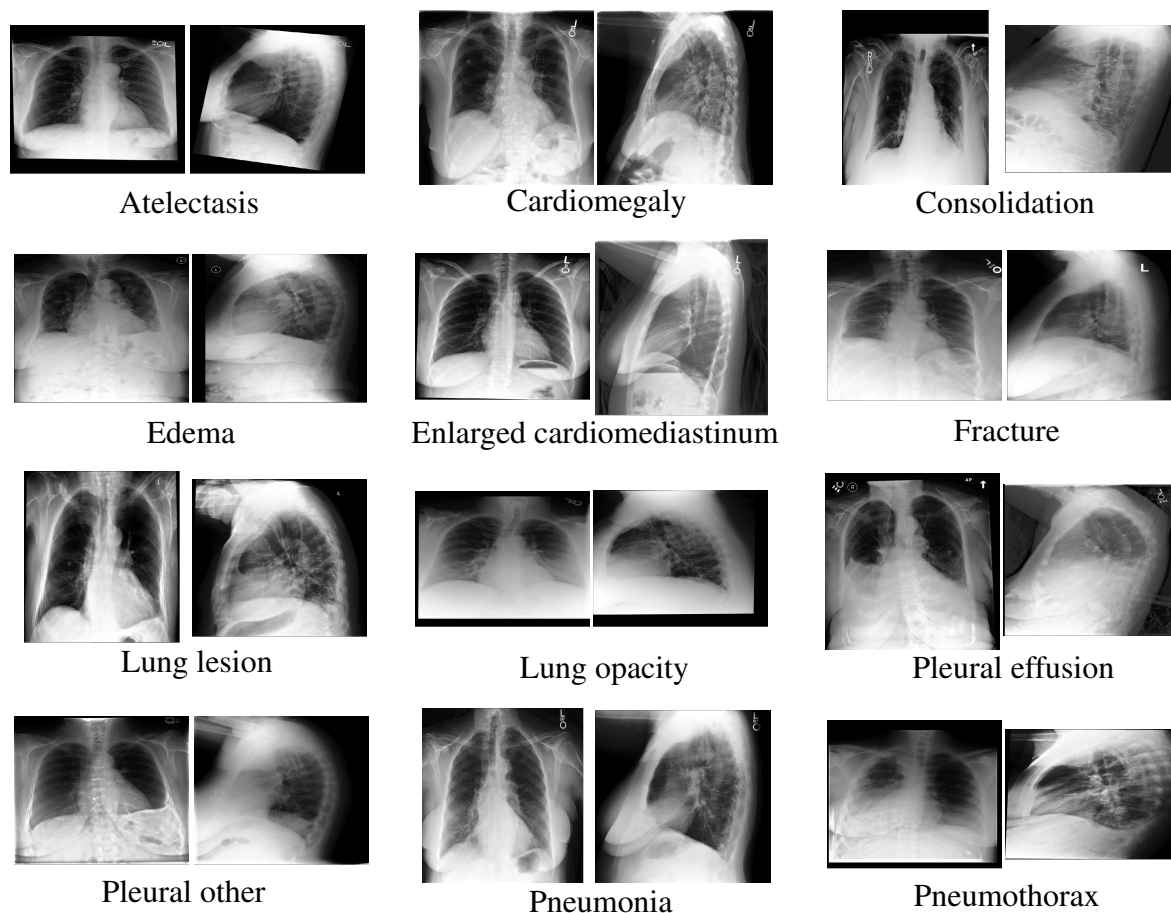


FIGURE 4.2. Examples of 12 Thoracic Diseases from MIMIC-CXR Dataset. Each disease is associated with frontal and lateral views of CXRs [146].

an applied probability of 0.5. Note that no vertical flips, rotations, zooms, or warps were done to the images. In addition, uncertain and unknown labels were dropped.

4.2.2 Structure overview

The task of detecting thorax diseases in CXRs was divided into 12 sub-tasks, where each task considered the presence and absence of a specific disease. This simplified the automated CXR interpretation problem to a binary classification task. Combining binary CXR labels and CNN showed promise in a tunable classifier that performed similar to expert radiologists [207].

TABLE 4.1. The MIMIC-CXR Dataset with 12 Labeled Pathologies. We counted the numbers of positive and negative observations in 10% of the dataset.

Pathology	Positive	(%)	Negative	(%)
Enlarged Cardiom.	1,019	2.8	35,367	97.19
Cardiomegaly	6,932	18.79	29,951	81.2
Airspace Opacity	7,582	20.42	29,542	79.57
Lung Lesion	1,060	(2.82	36,472	97.17
Edema	3,964	11.06	31,859	88.93
Consolidation	1,410	3.8	35,634	96.19
Pneumonia	2,738	7.83	32,202	92.16
Atelectasis	6,356	17.54	29,876	82.45
Pneumothorax	1,523	4.05	36,059	95.94
Pleural Effusion	7,869	21.34	28,994	78.65
Pleural Other	425	1.13	37,132	98.86
Fracture	805	2.13	36,829	97.86

TABLE 4.2. Data Augmentation for the CXRs. We applied a list of transforms parameters to the trained images.

Parameter	Value
Size	224
Flip (horizontally)	TRUE
Lighting	0.3
Affine	0.5

TABLE 4.3. The AUC per Epoch for Training the ResNet-50 CNN. This model detects cardiomegaly using CXRs of 299×299 or 224×224 pixels of chest X-rays.

Image Size	Epoch								Avg AUC per Epoch
	1	2	3	4	5	6	7	8	
299	0.565	0.733	0.758	0.791	0.798	0.804	0.804	0.807	0.757
224	0.725	0.733	0.747	0.785	0.793	0.799	0.802	0.802	0.773

Among the proposed variations of ResNet layers (i.e., 34, 50, 101, 152, and 1,202), we adopted the popular ResNet-50 network, which consists of 49 convolution layers and ends with one fully connected layer. Equation (4.1) defines the last output of residual unit x_l , where $F(x_{l-1})$ is the generated output after performing the convolution operations, batch

normalization and activation function on x_{l-1} . Importantly, we use cyclical learning rates to enhance performance by decreasing the number of epochs required to accomplish the accuracy threshold. For each binary label problem, ResNet-50 was used as the baseline CNN architecture in the three main training stages (Fig. 4.3).

$$x_l = F(x_{l-1}) + x_{l-1} \quad (4.1)$$

4.2.3 Training stages

In the first stage, the pre-trained ResNet-50 with the default fastai [105] hyperparameter values was trained for three epochs. That meant setting all layers to frozen, excluding the final dense layer and examining each X-ray three times. In other words, the first stage embraced the transfer learning approach to train faster with a model that was already trained to recognize 1,000 categories of things in ImageNet. At the end of stage 1, the model's weights were saved.

In the second stage, the whole model was trained again for one epoch by unfreezing the layers and using the fit-one-cycle method. The objective of this stage was to observe the model's performance to reduce the training time and increase the accuracy. If the AUC was decreased at the end of this training stage, the stage 1 weights were reloaded.

In the third stage, the whole model was trained again for four epochs using the optimal LR finder. The LR was set by default to about $1e - 3$ in stage 1 and changed manually to a range of lower LRs ($1e - 6$ to $1e - 4$) in stage 3. Figure 4.4 illustrates the plotted LR after the first and second stages of the model, where the red dots in the graphs indicate the steepest gradient points. Using different LRs for each layer at this stage was in line with the discriminate fine-tuning technique to tune each layer with various LRs. In this case, the model's parameters, θ , and the LR, η , were split into $\{\theta^1, \dots, \theta^L\}$ at time step "t" and $\{\eta^1, \dots, \eta^L\}$, respectively, where "L" is the number of layers. This updated version of the regular Stochastic Gradient Descent (SGD) with discriminative fine-tuning was defined as in Eq. (4.2), where $\nabla_{\theta^l, j}$ is the gradient of the model's objective function.

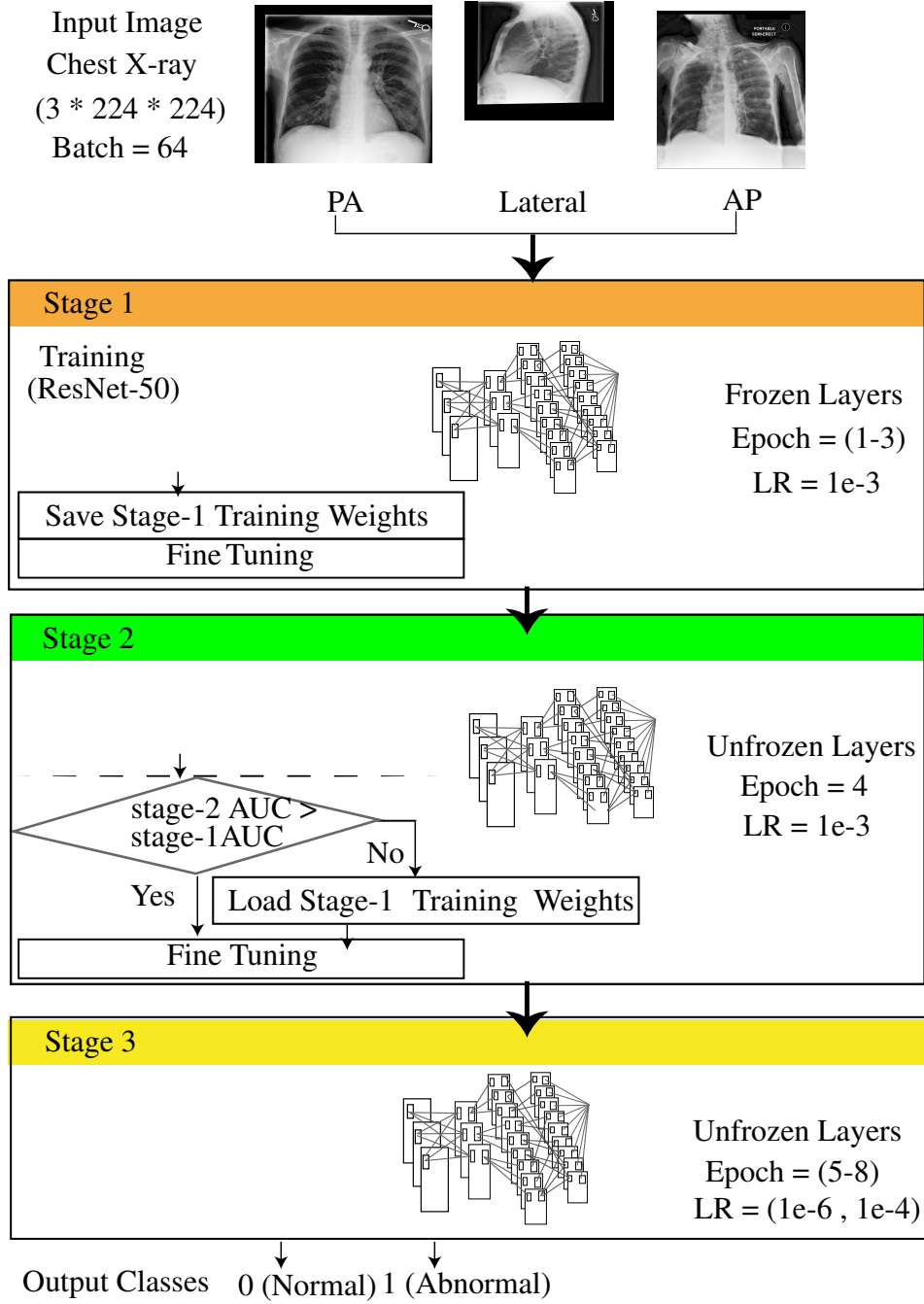


FIGURE 4.3. Overall Illustration of MultiViewModel [146].

$$\theta_t^l = \theta_{t-l}^l - \eta^l \times \nabla_{\theta^l} j(\theta) \quad (4.2)$$

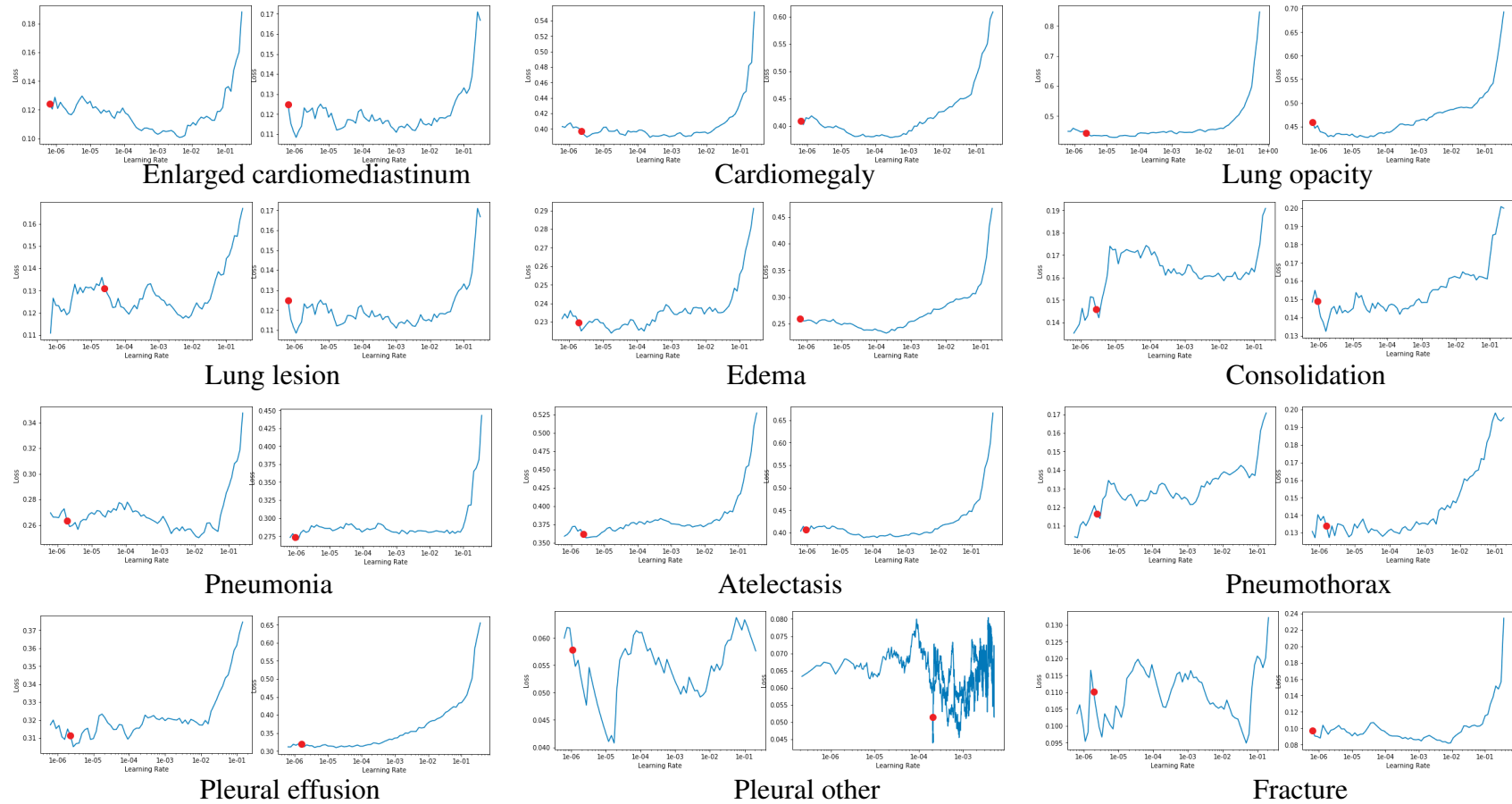


FIGURE 4.4. Fluctuated LR. Per pathology, the plot on the right represents the LR after the stage 1 training, and the plot on the left shows the LR after the stage 2 training. Note that the x-axis represents what happened as the LR increased, and the y-axis indicates what the loss was (color figure online) [146].

For a more precise classification, the ResNet-50 was fine-tuned by batch normalization and weight optimization. Tajbakhsh et al. [208] proved that training a fine-tuned CNN model takes substantially less time and gives better accuracy than a CNN model trained from scratch. This is because the weights in the fine-tuned CNN model are initialized to specific values learned from previous knowledge. Furthermore, the one-cycle policy was utilized to achieve high classification performance with only eight epochs of training. Previous research has demonstrated the one-cycle policy's effectiveness, showing that training a model for 20 epochs with this policy achieves a similar performance to training the model for 100 epochs without this policy [209]. After each training epoch, we printed the training loss, validation loss, and the AUC matrix on the validation set to examine the model's performance using an unseen dataset, where lower losses indicated better model performance.

4.3 Experiment

The training algorithms were evaluated with 12 pathologies: enlarged cardiomeastinum, cardiomegaly, airspace opacity, lung lesion, edema, consolidation, pneumonia, atelectasis, pneumothorax, pleural effusion, pleural other and fracture. We used the PyTorch software [103], fastai library, n1-highmem-8 (8 vCPUs, 52 GB memory) machine and 4 x NVIDIA Tesla P4 Graphics Processing Units (GPUs). This was in accordance with Coleman et al.'s work [210], which demonstrated how the time per epoch for the ResNet-50 architecture scale was much better when training it on multiple GPUs. Table 4.4 records the time per epoch for training the ResNet-50-based model to detect cardiomegaly using different numbers of GPUs, where parallel training on four GPUs reduced the training time by around 20 minutes.

Our proposed MultiViewModel is available in Appendix (B), and more details about the implementation of this model can be found on <https://github.com/MaramMonshi/MultiViewModel>.

4.4 Results and discussion

Table 4.5 shows the AUC results of each pathology computed on the validation set for each of the eight training epochs. It can be seen that the detection performance for each pathology

TABLE 4.4. Time per Epoch for Training the ResNet-50 CNN. This model detects cardiomegaly using a single NVIDIA Tesla P4 GPU or four NVIDIA Tesla P4 GPUs in a parallel training. Note that the batch size was set to 64 images, and the image size was set to 224 pixels.

No. of GPUs	Epoch								Avg. Time per Epoch (min)
	1	2	3	4	5	6	7	8	
1	32:42	32:26	32:36	34:34	33:40	33:52	33:58	34:00	33:28
4	13:32	12:54	13:01	13:05	13:07	13:08	13:07	13:06	13:07

TABLE 4.5. The Compression of the AUC Scores in each Epoch. We trained each pathology for eight epochs.

Pathology	Epoch							
	1	2	3	4	5	6	7	8
Enlarged Cardiomegaly	0.670	0.694	0.700	0.544	0.702	0.705	0.708	0.710
Cardiomegaly	0.725	0.733	0.747	0.785	0.793	0.799	0.802	0.802
Airspace Opacity	0.621	0.687	0.694	0.712	0.730	0.730	0.733	0.737
Lung Lesion	0.520	0.638	0.612	0.638	0.651	0.688	0.730	0.729
Edema	0.816	0.848	0.857	0.887	0.892	0.894	0.896	0.897
Consolidation	0.748	0.758	0.769	0.778	0.788	0.797	0.797	0.799
Pneumonia	0.556	0.531	0.545	0.497	0.550	0.585	0.580	0.587
Atelectasis	0.706	0.706	0.743	0.827	0.830	0.835	0.837	0.838
Pneumothorax	0.710	0.786	0.817	0.839	0.853	0.862	0.868	0.860
Pleural Effusion	0.837	0.869	0.881	0.891	0.903	0.906	0.905	0.899
Pleural Other	0.585	0.637	0.676	0.533	0.707	0.736	0.739	0.727
Fracture	0.546	0.563	0.576	0.606	0.636	0.648	0.607	0.741
Average	0.670	0.704	0.718	0.711	0.753	0.765	0.766	0.777

fluctuated over the epochs. For the individual training epochs, the eighth unfrozen epoch achieved a higher average AUC (0.777) compared to the first (0.670), second (0.704), third (0.718), fourth (0.711), fifth (0.753), sixth, (0.765) and seventh (0.766) epochs. Compared with stage 1 (epochs 1–3) and stage 2 (epoch 4), stage 3 (epochs 5–8) resulted in larger AUC values for all pathologies. This difference was likely due to the discriminative LRs in the third stage of training.

Furthermore, we followed the one-cycle training technique [205] to make ResNet-50 converge faster based on LR. The one-cycle training enabled the MultiViewModel to use higher LR than other types of training to train faster and overfit less by skipping over the sharp local

minima and ending up in a smother part of the loss. Smith and Topin [209] pointed out that faster training can be achieved by training with maximum LR and named this phenomenon "super-convergence."

Table 4.6 compares the pathology AUC results between our proposed model and the DualNet architecture using the MIMIC-CXR dataset. We employed 10% of the dataset using all available frontal and lateral views of the CXRs. DualNet, however, considered a combination of PA and lateral views as well as a composite of AP and lateral views. In five out of seven overlapped pathologies, our model performed better than both DualNet models. Overall, it can be seen that average AUC was higher for our MultiViewModel classifiers (0.779), compared to both the PA-lateral (0.722) and AP-lateral (0.677) classifiers.

However, the DualNet model performed better than our proposed MultiViewModel in detecting pneumonia and cardiomegaly using PA and lateral views. The combination of these two views, PA and lateral, is the standard chest examination, where a patient can stand in front of an X-ray source, and has better image quality than the PA views [211]. Pezzotti [212] recommended that the AP view should be reserved for very ill patients who cannot stand because the heart and other structures in the anterior part of the chest look bigger in the AP view. Thus, the AP view may lead to the false detection of several CXR findings, such as cardiomegaly [213]. Since MultiViewModel used the AP view in training, it had lower performance in detecting pneumonia and cardiomegaly than the DualNet model.

In the DualNet model, the CXR labels were extracted from the associated radiology reports using an open source tool developed by the National Institute of Health (NIH), the NegBio labeler [121]. This tool was used to annotate the popular ChestX-ray14 dataset. In contrast, our model followed the labels publicly released by Johnson et al. [113] that utilized a different open source tool by the Stanford Machine Learning Group, the CheXpert labeler. Although the labeling algorithm of CheXpert was built upon the work of NegBio, it achieves a higher F1 score. Hence, our model is trained on better-annotated CXRs than the DualNet model. Interestingly, we obtained improved results compared to those achieved by DualNet using smaller image sizes of 224×224 pixels instead of 512×512 pixels.

TABLE 4.6. The Compression of AUC Scores. The DualNet model used an older limited released version of the MIMIC-CXR dataset. Our model used 10% of the publicly released version of the dataset. Note that we ignored uncertain and unknown labels.

Pathology	DualNet [2]		Our
	PA + Lateral	AP + Lateral	MultiViewModel
Enlarged Cardiom.	-	-	0.710
Cardiomegaly	0.840	0.755	0.802
Airspace Opacity	-	-	0.737
Lung Lesion	-	-	0.730
Edema	0.734	0.749	0.897
Consolidation	0.632	0.623	0.799
Pneumonia	0.625	0.593	0.587
Atelectasis	0.766	0.671	0.838
Pneumothorax	0.706	0.621	0.868
Pleural Effusion	0.757	0.733	0.906
Pleural Other	-	-	0.739
Fracture	-	-	0.741
Average	0.722	0.677	0.779

Nevertheless, the MIMIC-CXR dataset has the largest number of open source X-ray images to date; the class labels in the training set are noisy because they were mined by Natural Language Processing (NLP) tools, rather than by experienced radiologists. Figure 4.5 visualizes the most common X-rays incorrectly predicted by our model with heatmaps using the activations of the wrongly predicted class. In addition, the positive negative subsets ratio was highly imbalanced in the enlarged cardiomeastinum, lung lesion, consolidation, pneumothorax, pleural other, and fracture sets (Table 4.1). However, our model’s AUC for each of these pathologies was above 0.7 (Table 4.6).

4.5 Summary and conclusion

In this chapter, ResNet-50 CNN-based stage-wise models were proposed to detect 12 thorax diseases in 10% of the largest CXR dataset to date, the MIMIC-CXR dataset. The absolute labeling performance with an average weighted AUC of 0.779 is encouraging, since we

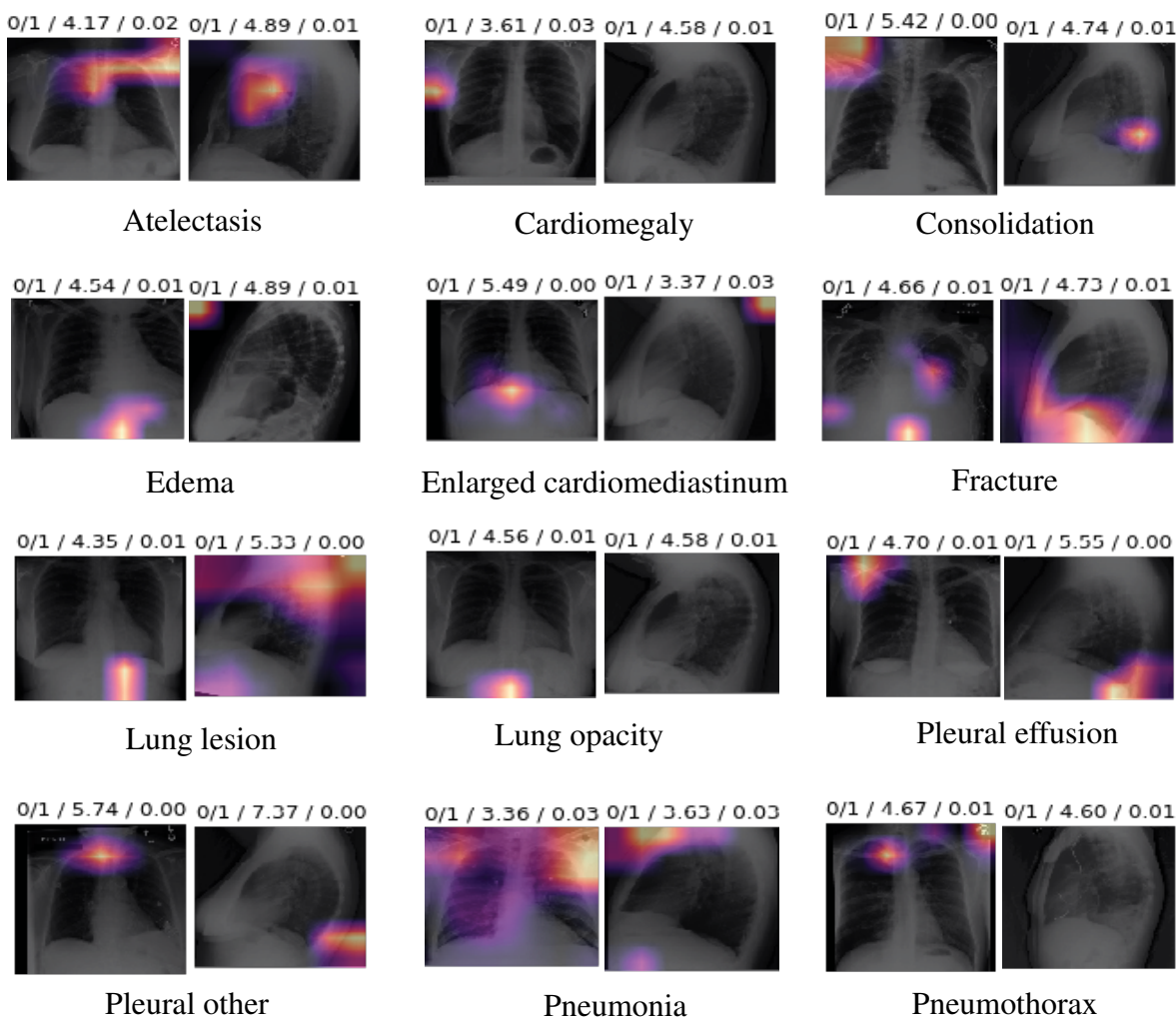


FIGURE 4.5. Examples of the Most Confused CXRs with Heatmaps. Each image was associated with the prediction, actual, loss and probability values after the stage 1 training, where 0 and 1 represent negative and positive pathology respectively [146].

used only a subset of the available CXRs. In future work, we plan to improve our MultiViewModel's performance through utilizing common image-based classification techniques, particularly data augmentation. Importantly, we will incorporate useful information from the free-text radiology reports, such as patients' history and clinical records, to accurately recognize the presence and absence of thorax diseases.

Multi-Label Image Classification¹

Binary chest radiography classifiers have been widely proposed in the literature, including our MultiViewModel in the previous chapter, due to the evolution of Deep Learning (DL) and the availability of large Chest X-ray (CXR) datasets. However, these automatic classifiers neglect label co-occurrence and interdependency in CXRs and fail to make full use of accelerators, resulting in inefficient and computationally expensive models. This chapter first studies the effect of CXR image formats, variations of the Densely Connected Convolutional Network (DenseNet) [86] architecture, and parallel training on chest radiography multi-label classification tasks. Then, we propose Xclassifier, an efficient multi-label classifier that trains an enhanced DenseNet with a blur pooling framework to classify CXRs based on 14 predefined labels.

5.1 Introduction

Chest X-rays are of great importance for clinical diagnosis as they contain rich relationship information among pathologies, such as label co-occurrence of multiple observations [214]. The availability of large public CXR datasets [13][110][113][115] and the evolution of DL offer an optimal solution for the multi-label chest radiography classification problem. Consequently, many models have recently been proposed for applications in classifying chest radiographs [34][88][108][146]. We described these models in Chapter 2 (Literature

¹The content in this chapter has been published in the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), "Distributed Deep Learning for Multi-Label Chest Radiography Classification", Monshi, M., Poon, J., Chung, V. (2022).

Review), section 2.6.2. However, these models did not capture the label dependencies in chest radiographs, and effectively accomplishing this task is still a challenge [215].

On the computation side, the computational power grows tremendously with the introduction of a state-of-the-art Graphics Processing Unit (GPU) such as NVIDIA A100 [216] or NVIDIA V100 [217], but on-device memory is often constrained. The NVIDIA A100 GPU is the new generation of accelerator GPUs but is still not supported on all platforms. Parallel training, however, is performing multiple processes on devices of a single machine or multiple machines. As public chest radiography datasets and the number of DL layers increase, one GPU quickly becomes insufficient to accelerate neural network training. However, evaluation of these techniques in real-world applications, such as classifying CXRs, are limited.

Training a DL model in parallel trains a model across multiple GPUs to speed up neural network training. This training approach is essential for training the large public CXRs that have been recently introduced one after another. For example, ChestX-Ray14 [110], Pathology Detection in Chest radiographs (PadChest) [115], CheXpert [13], and Medical Information Mart for Intensive Care Chest X-ray (MIMIC-CXR) [113] have 112,120, 168,861, 224,316, and 377,110 images, respectively.

Parallel training can be achieved by Data Parallel (DP) or Distributed Data Parallel (DDP) [218] techniques. The DP technique means performing one process (i.e., training a DL model) on multiple devices (i.e., multiple GPUs) of a single machine by distributing batches of the data on the available GPUs. Although a batch size can be large in the DP technique, the processing time is long due to the limitation of using one process. Meanwhile, the DDP technique enables each device to independently conduct one process on a portion of the training dataset [218].

Furthermore, existing chest radiography classifiers' performances can be improved by leveraging label co-occurrence [215], selecting the optimal radiographs format [219], and training with an efficient approach. In studying previous methods used to solve these issues, it can be noted that the existing literature rarely discusses the efficiency of chest radiography classifiers.

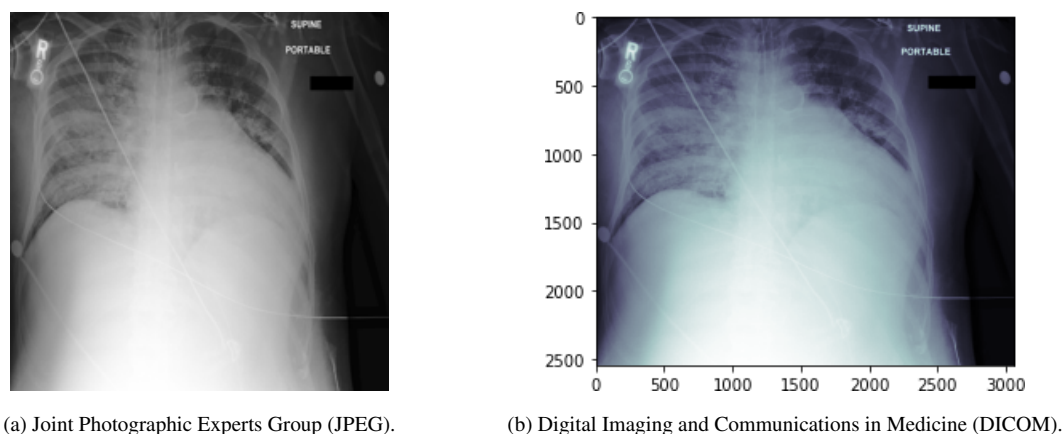


FIGURE 5.1. The CXR Image Formats [152].

5.1.1 Contributions

Our contributions can be outlined as follows: Regarding the multi-label CXR classification task, we quantify the value of the optimal image format, study parallel DL in accelerating neural network training, and compare the performance of variations of DenseNet-121. Then, we propose the Xclassifier, an efficient and accurate multi-label CXR classifier, based on an enhanced DenseNet-121 framework with antialiasing blur pooling and parallel training.

The most common file format used to store medical imaging data for patient medical scans, such as CXRs, Computed Tomography (CT) scans, and Magnetic Resonance Imaging (MRI) scans, is Digital Imaging and Communications in Medicine (DICOM) [38]. However, most existing DL models in medical image prediction utilize the Joint Photographic Experts Group (JPEG) format due to the limitations of Compute Engine machines. Figure 5.1 shows an example of DICOM and JPEG CXRs. Recently, researchers have started to extract image categories from DICOM metadata (i.e., study and image descriptions) and map them to the World Health Organization manual of diagnostic imaging [220]. However, to the best of our knowledge, there has not been any comparison between DICOM and JPEG formats on the performance of multi-label classifiers for chest radiographs using DL.

5.2 Proposed Xclassifier model

5.2.1 Data preparation

The MIMIC-CXR and CheXpert datasets were used in this study with more than 500,000 labeled chest radiographs, as described in Chapter 2 (Literature Review), section 2.4 CXR datasets. We utilized 356,225 CXRs from MIMIC-CXR, and 212,498 of the low-resolution images from CheXpert. Unlike the CheXpert competition, the task of this study was to detect 14 observations simultaneously rather than five. We explicitly examined the dependencies between labels on the MIMIC-CXR dataset in Table 5.1. The table illustrates, for instance, that 37% of the cardiomegaly-labeled CXRs were also labeled pleural effusion. Table 5.2 represents the label co-occurrence in the CheXpert dataset. For instance, 43% of the atelectasis-labeled CXRs were also labeled lung opacity. The dependencies between the labels in each dataset emphasized the importance of labeling the datasets in a multi-label method rather than a single label method.

In both datasets, we converted uncertain and missing values to negative, following the U-Zeros model [13]. We ensured that each CXR had at least one positive label because a positive “no finding” label indicated the absence of all pathologies. In addition, we randomly shuffled the CXRs into three splits, i.e., 80% for training, 10% for validation, and 10% for testing, using a fixed random seed of 42.

5.2.2 Multi-label classifier

Data augmentation: For the data augmentation, we squished each CXR to 240×240 pixels (i.e., resized each CXR by squishing it on the horizontal axis), rotated it by 20° , zoomed in by 1.2 scale, warped it by 0.2 magnitude, lightened it by 0.3 scale, and normalized it. These data augmentation parameters increased the accuracy of detecting abnormalities from CXRs based on the extensive experiment results in Chapter 6. Importantly, we have only applied data augmentation on the training set, where the validation and test sets always had the original images.

TABLE 5.1. Positive Label Co-occurrence of MIMIC-CXR.

Label	% of all data	% of label co-occurrence													
		At	Ca	Co	Ed	EC	Fr	LL	LO	NF	PE	PO	Pa	Px	SD
Atelectasis (At)	18	100	29	5	13	5	2	3	31	0	48	1	8	6	39
Cardiomegaly (Ca)	18	28	100	5	23	4	2	2	25	0	37	1	8	4	41
Consolidation (Co)	4	22	23	100	21	5	2	6	27	0	50	1	22	4	44
Edema (Ed)	10	24	40	8	100	4	1	2	29	0	51	1	11	2	37
Enlarged Cardiom. (EC)	3	32	23	7	14	100	3	6	33	0	36	2	7	8	45
Fract (Fr)	2	21	19	2	6	4	100	3	19	0	21	3	4	9	23
Lung Lesion (LL)	3	18	13	8	6	5	2	100	46	0	26	3	11	4	18
Lung Opacity (LO)	21	27	21	5	14	4	2	7	100	0	32	2	17	4	31
No Finding (NF)	40	0	0	0	0	0	0	0	0	100	0	0	0	0	10
Pleural Effusion (PE)	22	41	31	10	24	5	2	4	31	0	100	1	9	6	41
Pleural Other (PO)	1	15	25	4	9	6	7	8	39	0	26	100	10	5	25
Pneumonia (Pa)	7	20	18	12	15	3	1	5	48	0	26	1	100	1	21
Pneumothorax (Px)	4	28	17	5	6	6	5	3	21	0	33	1	3	100	54
Support Devices (SD)	24	31	31	8	16	5	2	2	28	16	37	1	7	9	100

TABLE 5.2. Positive Label Co-occurrence of CheXpert.

Label	% of all data	% of label co-occurrence													
		At	Ca	Co	Ed	EC	Fr	LL	LO	NF	PE	PO	Pa	Px	SD
Atelectasis (At)	16	100	12	6	27	5	4	3	43	0	49	1	2	9	60
Cardiomegaly (Ca)	13	14	100	5	43	7	3	2	48	0	44	1	2	3	58
Consolidation (Co)	7	14	10	100	21	4	3	5	38	0	50	2	7	5	52
Edema (Ed)	25	17	22	6	100	4	2	2	53	0	51	1	2	3	64
Enlarged Cardiom. (EC)	14	18	6	20	20	100	6	5	48	0	36	2	1	7	52
Fract (Fr)	4	14	9	4	11	7	100	4	40	0	27	3	2	12	40
Lung Lesion (LL)	4	11	7	8	9	6	4	100	58	0	36	3	5	9	35
Lung Opacity (LO)	50	13	12	5	26	5	3	5	100	0	49	2	4	9	58
No Finding (NF)	11	0	0	0	0	0	0	0	0	100	0	0	0	0	39
Pleural Effusion (PE)	41	19	14	9	31	5	3	4	61	0	100	1	2	8	61
Pleural Other (PO)	2	11	9	9	5	8	9	53	0	26	100	4	7	39	
Pneumonia (Pa)	3	10	8	17	20	3	2	8	67	0	29	2	100	2	29
Pneumothorax (Px)	9	16	4	4	8	4	5	4	47	0	34	1	1	100	60
Support Devices (SD)	55	17	13	7	29	5	3	3	53	8	46	1	2	10	100

Convolutional neural network architecture: Xclassifier was based on DenseNet due to the success of this architecture in recent classification models using CXR datasets [108][144][215]-[221][222]. DenseNet utilizes dense blocks to connect all layers directly with each other by matching feature map sizes. As demonstrated in Fig. 5.2, each layer in this CNN passed on its own feature maps to all successive layers and collected additional inputs from all prior layers to maintain the feed-forward nature.

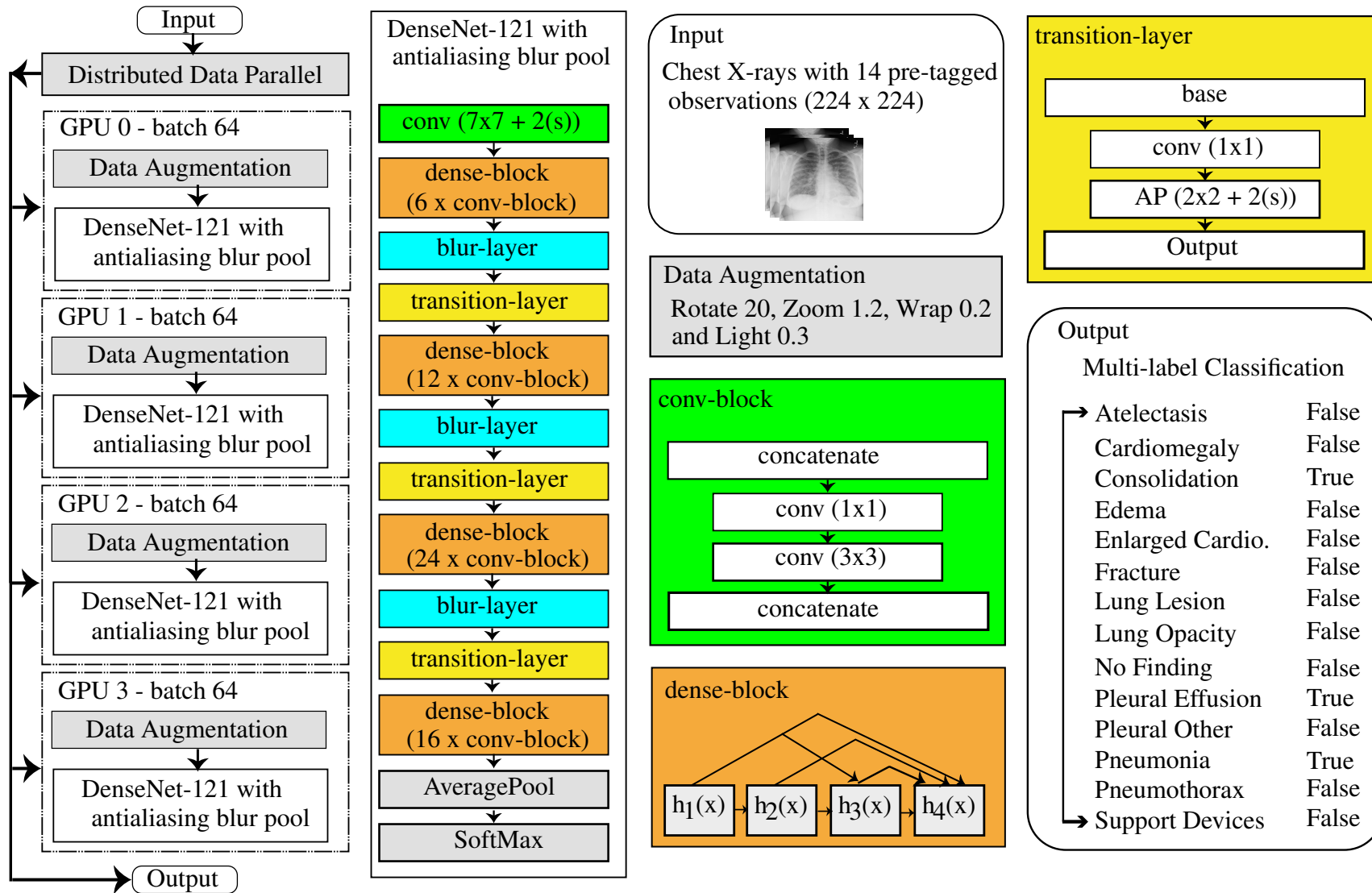


FIGURE 5.2. The Xclassifier Structure [152].

TABLE 5.3. The DenseNet-121 Variations Models and Training Performances. We used the full MIMIC-CXR dataset and trained for 10 epochs.

Model	Description	Accuracy	AUC
DenseNet-121	Single 7x7 convolution layer with no antialiasing layer	90.69	81.34
DenseNet-121d	Three 3x3 convolution layers with no antialiasing layer	90.73	81.28
DenseNetblur-121d	Three 3x3 convolution layers with antialiasing blur pool	90.80	81.96

Equation (5.1) represents the dense connectivity where $[x_0, x_1, \dots, x_{\ell-1}]$ donates the concatenation of the feature maps produced in layers $0, 1, \dots, \ell - 1$. Each DenseNet architecture consisted of four dense blocks with a varying number of layers. Xclassifier had six, 12, 24, and 16 layers in the four dense blocks as in DenseNet-121. We did not use the deeper architectures of DenseNet (i.e., 161, 169, 201, and 264) because increasing the number of DenseNet hidden layers would not improve the CXR classification performance [88].

$$x_\ell = H_\ell([x_0, x_1, \dots, x_{\ell-1}]) \quad (5.1)$$

Anti-aliasing and subsampling: Before each downsampling step in DenseNet, we inserted a blur kernel, $m \times m$, as an antialiasing filter. We found that this minor modification increased the CXR classification accuracy, as illustrated in Table 5.3. Additionally, previous research has shown that modifying the backbone of several CNN architectures by adding a blur kernel can increase the accuracy of ImageNet classification [223]. We applied the anti-aliasing, as depicted in Eq. (5.2), at stride 2 of DenseNet. Note that $BlurPool_{m,s}$ denotes the image processing function that combines blurring and subsampling, where k is the kernel and s is the stride.

$$Relu \circ Conv_{k,s} \rightarrow BlurPool_{m,s} \circ Relu \circ Conv_{k,1} \quad (5.2)$$

Fine-tuning: To fine-tune Xclassifier, we adopted the fit-one-cycle policy [205] and discriminative learning rates [195]. This policy of cyclical learning rates worked as a regularization technique to achieve faster and better training and hence kept the network from overfitting.

NVIDIA-SMI 460.73.01 Driver Version: 460.73.01 CUDA Version: 11.2									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	BCC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.		
0	Tesla V100-SXM2...	Off	00000000:00:04:0	Off	37%	Default	0		
N/A	49c	PO	149w / 300W	4238MiB / 16160MiB			N/A		
1	Tesla V100-SXM2...	Off	00000000:00:05:0	Off	30%	Default	0		
N/A	46c	PO	112w / 300W	3994MiB / 16160MiB			N/A		
2	Tesla V100-SXM2...	Off	00000000:00:06:0	Off	19%	Default	0		
N/A	47c	PO	107w / 300W	4054MiB / 16160MiB			N/A		
3	Tesla V100-SXM2...	Off	00000000:00:07:0	Off	39%	Default	0		
N/A	47c	PO	99w / 300W	3936MiB / 16160MiB			N/A		

NVIDIA-SMI 460.73.01 Driver Version: 460.73.01 CUDA Version: 11.2									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	BCC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.		
0	Tesla V100-SXM2...	Off	00000000:00:04:0	Off	94%	Default	0		
N/A	48c	PO	171w / 300W	12710MiB / 16160MiB			N/A		
1	Tesla V100-SXM2...	Off	00000000:00:05:0	Off	92%	Default	0		
N/A	47c	PO	90w / 300W	12730MiB / 16160MiB			N/A		
2	Tesla V100-SXM2...	Off	00000000:00:06:0	Off	95%	Default	0		
N/A	47c	PO	88w / 300W	12068MiB / 16160MiB			N/A		
3	Tesla V100-SXM2...	Off	00000000:00:07:0	Off	94%	Default	0		
N/A	48c	PO	67w / 300W	12638MiB / 16160MiB			N/A		

Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	Usage
ID	ID	ID					
0	N/A	N/A	9951	C	/opt/conda/bin/python	4235MiB	
1	N/A	N/A	9951	C	/opt/conda/bin/python	3991MiB	
2	N/A	N/A	9951	C	/opt/conda/bin/python	4051MiB	
3	N/A	N/A	9951	C	/opt/conda/bin/python	3933MiB	

Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	Usage
ID	ID	ID					
0	N/A	N/A	29057	C	/opt/conda/bin/python	12707MiB	
1	N/A	N/A	29058	C	/opt/conda/bin/python	12729MiB	
2	N/A	N/A	29059	C	/opt/conda/bin/python	12065MiB	
3	N/A	N/A	29060	C	/opt/conda/bin/python	12635MiB	

(a) Data Parallel (DP).

(b) Distributed Data Parallel (DDP).

FIGURE 5.3. Visualizing Parallel Training Approaches. We used four Tesla V100 GPUs and trained DenseNetblur-121d for multi-label classification tasks [152].

Distributed data parallel: With the DDP technique [218], we could use a large batch size of 64 images for each of the four GPUs to accelerate the convergence. In every training iteration, the single-device memory was frequently above 91% during backward propagation, where each GPU independently performed one copy of the training on part of the dataset. Figure 5.3b captures a live example of the Xclassifier training job using four Tesla V100-SXM2-16 GB GPUs. It shows the normalized GPU utilization of both the computing core and memory usage.

5.3 Experiment

For distributed DL, we used the PyTorch DDP [218], PyTorch image models (timm) [224], the Fastai v2 library [105], and an n1-highmem-32 (32 vCPUs, 208 GB memory) machine with four NVIDIA Tesla V100 GPUs. We used a batch size of 64 for each of the four GPUs and trained the model for 30 epochs.

Our proposed Xclassifier is available in Appendix (B), and more details about the implementation of this model can be found on <https://github.com/MaramMonshi/Xclassifier>.

TABLE 5.4. Image Formats for the CXRs and Training Performance. We used 10% of the MIMIC-CXR and trained ResNet-18 for 10 epochs.

Chest x-ray format	Accuracy	AUC	Avg. time per epoch (min)
DICOM	89.40	80.02	111
JPEG	89.58	81.57	6

5.4 Results and discussion

CXR image format: A comparison of the accuracy and Area Under the Receiver Operating Characteristic Curve (AUC) values for the DICOM versus JPEG format for the multi-label classification task is demonstrated in Table 5.4. Although the DICOM format is more readily applicable than the JPEG format to clinical practice, it did not improve the automated neural network accuracy here. In fact, it took significantly more time to train with the DICOM files (i.e., 111 minutes per epoch) than their JPEG counterparts (i.e., six minutes per epoch), using 10% of the MIMIC-CXR dataset. Therefore, we decided not to train the DICOM files any further.

This ablation study suggested that JPEG images are more efficient than their DICOM counterparts in the multi-label classification task using DL, as documented in Table 5.4. However, in practice, radiologists use a finer resolution of CXR, i.e., the DICOM format, to detect multiple observations carefully. Therefore, for future work, we plan to investigate the use of DICOM images in detecting diseases with small and complex structures to understand our initial findings better.

DensNet-121 variations: A comparison of the accuracy and AUC values for DenseNet-121 versus DenseNet-121d versus DenseNetblur-121d for the multi-label classification task is shown in Table 5.3. DenseNet-121 with the blur pooling outperformed its variations, so we built the Xclassifier on top of this architecture. Due to the shift-variant nature of CNNs, anti-aliasing filters were used to increase the accuracy of Xclassifier.

Compared to the basic DenseNet-121 model, Xclassifier anti-aliased the stride layers of DenseNet-121 and fine-tuned DenseNet-121 on the MIMIC-CXR dataset using the fit-one-cycle

TABLE 5.5. Training Approaches and Training Performance. We used the NVIDIA V100 GPU.

Training Approach	Dataset	Accuracy	AUC	Avg. time per epoch (min)
Single GPU (1 x GPU)	CheXpert	88.09	78.55	16
Data parallel (4 x GPUs)	CheXpert	88.36	79.25	14
Distributed data parallel (4 x GPUs)	CheXpert	88.33	80.10	4
Data parallel (4 x GPUs)	MIMIC-CXR	90.27	80.97	181
Distributed data parallel (4 x GPUs)	MIMIC-CXR	90.31	81.76	54

policy and discriminative learning rates. These modifications to the basic DensNet-121 resulted in faster and better training and kept the network from overfitting.

This ablation study compared different variations of DenseNet-121, as recorded in Table 5.3, and concluded that adding an anti-aliasing filter to CNNs may enhance the performance of multi-label CXR image classifiers due to the shift-variant nature of CNNs. However, the effectiveness of adding an anti-aliasing filter to other popular CNN architectures in the CXR image classification domain (e.g., ResNet [78] and EfficientNet [225]) should be investigated. We leave this anti-aliasing filter investigation as a future research direction.

Parallel training: A comparison of the average time per epoch for a single GPU versus DP versus DDP for the multi-label classification task using DenseNetblur-121d is illustrated in Table 5.5. The DDP technique was the best training approach for CheXpert in terms of time efficiency, providing a four-fold increase in speed over the single GPU, and a 1.14- to 3.35-fold increase in speed over the DP technique.

Benchmark: The proposed Xclassifier improved the multi-label classification performance by 0.70% AUC (84.10% vs. 83.40%) on the MIMIC-CXR and by 3.39% AUC (83.89% vs. 80.50%) on the CheXpert (refer to Table 5.6). As Xclassifier depended on the DDP of DenseNet blur 121, it allowed the CNN layers to be deeper, more accurate in learning label co-occurrence, and more efficient to train. Figure 5.4 represents a sample of the labels correctly produced by Xclassifier.

Label co-occurrence learning is beneficial in the field of multi-label CXR classification [215][226]. Our research showed that multi-label classifiers boost training performance in

TABLE 5.6. Comparing Xclassifier with the Benchmark.

Multi-label classifier	Dataset	Accuracy	AUC
Latent-space self-ensemble [111]	CheXpert	–	66.97
CheXclusion [112]	CheXpert	–	80.50
Xclassifier	CheXpert	89.61	83.89
VSE-GCN [114]	MIMIC-CXR	–	72.10
CheXclusion [112]	MIMIC-CXR	–	83.40
Xclassifier	MIMIC-CXR	92.17	84.10

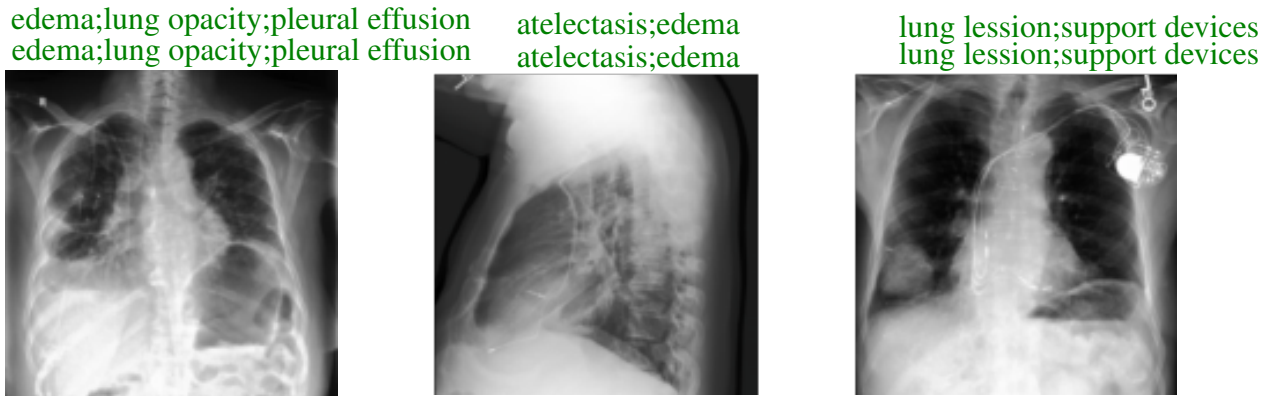


FIGURE 5.4. Correct Output Sample by Xclassifier [152].

terms of speed and accuracy compared to CXR binary classifiers, such as our MultiViewModel in the previous chapter (4). For example, the first CXR image in Fig. 5.4 has positive labels for edema, lung opacity, and plural effusion and negative labels for atelectasis, cardiomegaly, consolidation, enlarged cardiomeastinum, fracture, lung lesion, no finding, pleural other, pneumonia, pneumothorax, and support device. Such output labels would take at least 17 hours of training of the MultiViewModel (i.e., binary label classifier) with an average of 77.90 AUC, compared to less than 9 hours with an average of 84.10 AUC using Xclassifier (i.e., multi-label classifier). However, more advanced methods using interdependencies between CXR findings remain unexplored.

5.5 Summary and conclusion

We introduced Xclassifier, an efficient multi-label classifier that trains an enhanced DenseNet-121 framework with blur pooling to detect 14 observations from a CXR. It accomplished an ideal memory utilization and GPU computation and achieved 84.10% AUC on the MIMIC-CXR dataset and 83.89% AUC on the CheXpert dataset. Xclassifier used features of all complexity levels to handle label co-occurrence training. The DDP technique is a true data parallelism process. It is useful in performing multi-processes on devices of multiple machines but can also be used on devices of just a single machine as well.

For future work, we plan to investigate the use of DICOM images in detecting diseases with small and complex structures to offer a greater degree of understanding of our initial findings. In practice, radiologists use a finer resolution of CXRs, i.e., the DICOM format, and rely on additional information, such as the patients' electronic health records, to detect multiple observations. Furthermore, we plan to link patient data, such as age and gender, to the flattened layer to improve prediction.

Multi-Class Image Classification: COVID-19 Detection¹

As seen in the previous chapters, Chest X-ray (CXR) images coupled with Convolutional Neural Network (CNN) algorithms can speed up the diagnostic process of lung-related diseases. In this chapter, we detect a current pandemic lung disease named Coronavirus Disease 2019 (COVID-19) from CXRs through optimizing the CNN hyperparameters and the data augmentation in terms of the validation accuracy. We propose CovidXrayNet, a multi-class classifier that is based on EfficientNet and our optimization results to classify a CXR as being either "COVID-19," "normal," or "pneumonia."

6.1 Introduction

Coronavirus disease 2019, caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), became a global pandemic in less than four months after first appearing in December 2019 in Wuhan, China. It has since caused 620.30 million confirmed cases and over 6.54 million deaths worldwide as of October 13, 2022 [12]. It has caused devastating issues in public health and the global economy. Patients with COVID-19 may have one or more of the following symptoms: fever, cough, sore throat, headache, fatigue, muscle pain, and shortness of breath [55]. Early detection of positive COVID-19 cases is the most critical factor in slowing the spread of this pandemic.

The golden standard for diagnosing patients with COVID-19 is the Reverse Transcription Polymerase Chain Reaction (RT-PCR) testing, which detects SARS-CoV-2 through collected

¹The content in this chapter has been published in Computers in Biology and Medicine, "CovidXrayNet: Optimizing data augmentation and CNN hyperparameters for improved COVID-19 detection from CXR", Monshi, M., Poon, J., Chung, V., Monshi, F. (2021).

respiratory specimens of nasopharyngeal or oropharyngeal swabs [227]. However, RT-PCR testing is time-consuming and laborious, and shows poor sensitivity [228]. Alternatively, chest radiography imaging, including Computed Tomography (CT) or CXR, may be examined by a radiologist to inspect any visual indicators linked to SARS-CoV-2 [229]. While CT scans have greater detail, CXR images are more accessible, portable, and offer rapid triaging. Moreover, CXR imaging is more accessible in most healthcare systems than CT, which requires expensive equipment and maintenance. The portability of the CXR systems reduces the risk of COVID-19 transmission by performing the exams within the isolation room, which is not possible with fixed CT scanners. Importantly, CXRs allow for rapid triaging of suspected COVID-19 cases in most affected countries, such as the United States of America, Spain, and Italy, which have run out of both hospital capacity and RT-PCR testing supplies [230]. Combining laboratory results with radiological image features can speed up the process of COVID-19 detection.

Artificial Intelligence (AI) applications coupled with chest radiological imaging can speed the COVID-19 diagnosis process. Deep Learning (DL) in particular enables AI-based models to achieve accurate results without manual feature extraction [231]. For example, CNNs, which are a supervised DL approach, have recently gained popularity among the research community of AI in medicine. For COVID-19 detection from CXR images, CNNs produced the best classification accuracy compared to other classification techniques, such as Artificial Neural Network (ANN), Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) [232].

Typically, a CNN model is created by combining one or more of the following layers: a convolution layer, a pooling layer, and a fully connected layer. They extract features from the input, minimize the size for computational performance, and classify an image, respectively. Simultaneously, the CNN model adjusts its internal parameters to achieve a specific task, e.g., classifying CXRs [34][146]. The performance of such CNN models can be improved in various ways, such as by optimizing the data augmentation and CNN hyperparameters.

A growing number of research publications have demonstrated the compelling ability of DL CNNs to automatically detect COVID-19 from CXR images. They used public datasets

of COVID-19 CXRs as described in Chapter 2 (Literature Review), section 2.4.6. These COVID-19 datasets are constantly updated with new images added by researchers around the world. Nevertheless, none of these datasets provides complete metadata for all patients.

Table 6.1 summarizes the CNN-based models proposed in the literature, which can be grouped into binary classification (i.e., COVID-19 or normal) and multi-class classification (COVID-19, pneumonia, or normal) models. We analyzed these models in detail in Chapter 2 (Literature Review), section 2.6.

Table 6.2 and Table 6.3 outline the data augmentation and the CNN hyperparameters, respectively, in the recently proposed models. Nishio et al. [153] showed that combining multiple data augmentation techniques is more effective than only using one or not using any in detecting COVID-19 from CXRs. They utilized a random search [233] to select the optimal Visual Geometry Group VGG-16 [76] hyperparameters and data augmentation methods, including conventional method and mixup methods [143]. This resulted in an increase in their model's accuracy from 78.72% to 83.68%. However, this approach of hyperparameter tuning is hard to achieve with complex networks, such as EfficientNet [225], due to the large number of trainable parameters.

In terms of optimizing CNN hyperparameters, existing models use pre-trained architectures on ImageNet, Adam optimizer [234], epochs that range from 10 to 100, and a batch sizes of eight, 32, 64, or 128. Notably, several proposed architectures apply few arbitrary transformers to the X-rays based on random choices rather than well-justified motives. For instance, Ozturk et al. [151] applied the default values in the fastai v1 library. However, selecting the optimal CNN hyperparameters and data augmentation methods improves the robustness of CNN models [95].

TABLE 6.1. Models for Detecting COVID-19 from CXRs.

Classification	Model	Acc (%)	Repositories / Datasets	# of cases		
				COVID-19	Pneumonia	Normal
Binary	COVIDX-Net [148]	90.00	COVID-19 image data collection	25	–	25
	CovXNet [149]	97.40	Guangzhou Medical Center in China & Sylhet Medical College in Bangladesh	305	–	305
	ResNet-50 [150]	98.00	COVID-19 image data collection & Kaggle	50	–	50
	DarkCovidNet [151]	98.08	COVID-19 image data collection & ChestXray-14	125	–	500
Multi-class	VGG-16 [153]	83.68	COVID-19 image data collection & Radiological Society of North America (RSNA)	215	533	500
	DarkCovidNet [151]	87.02	COVID-19 image data collection & ChestXray-14	125	500	500
	CovXNet [149]	90.30	Guangzhou Medical Center in China & Sylhet Medical College in Bangladesh	305	305-Viral 305-Bacterial	305
	COVID-Net [118]	93.30	COVIDx	53	5526	8066
	MobileNet-v2 [154]	94.72	COVID-19 image data collection, Radiological Society of North America (RSNA), Radiopaedia, and Italian Society of Medical, Interventional Radiology (SIRM) & Kermany dataset	224	700	504
	CNN-SVM [155]	95.33	COVID-19 image data collection, COVID-19 radiography database & Kermany dataset	127	127	127

TABLE 6.2. Data Augmentation for Detecting COVID-19 from CXRs.

Model	Software	Norm.	Size	Flip	Rotate	Zoom	Light	Extra
VGG-16 [153]	Keras, Tensorflow	_	220*220	HORIZ	15	85-115%	_	shear transformation mixup: 0.1
DarkCovidNet [151]	fastai v1, Pytorch	yes	256*256	_	_	_	_	default values of fastai rescale: 1/255
CovXNet [149]	Keras, Tensorflow	yes	uniform	_	30	0.2	_	shift: 0.1
COVID-Net [118]	Keras, Tensorflow	yes	480*480	HORIZ	yes	yes	_	intensity shift
MobileNet v2 [154]	_	_	200*266	_	_	_	_	background: 1:1.5

TABLE 6.3. The CNN Hyperparameters for Detecting COVID-19 from CXRs.

Model	CNN	Pretrained	Optimizer	Learning Rate	Loss Function	Epoch	Batch
VGG-16 [153]	VGG-16	yes	Adam	1e-4	cross entropy	100	8
DarkCovidNet [151]	YOLO DarkNet-19	yes	Adam	3e-3	cross entropy	100	32
CovXNet [149]	CovXNet	yes	Adam	1e-3	cross entropy	70	128
COVID-Net [118]	COVID-Net	yes	Adam	2e-4 & lr policy	_	22	64
MobileNet v2 [154]	MobileNet v2	yes	Adam	_	_	10	64

6.1.1 Contributions

The main contribution of this study is the implementation of CovidXrayNet, which improves the detection rate of COVID-19 from CXRs by means of optimizing the data augmentation pipeline and CNN hyperparameters. To the best of our knowledge, CovidXrayNet is one of the first models to demonstrate the effects of data augmentation pipelines on CXR quality while also investigating several CNN hyperparameters. This in turn may significantly enhance the accuracy of CNNs in diagnosing COVID-19. In addition, we introduce COVIDcxr, a balanced and complete dataset that consists of CXRs and the associated tabular data.

In this chapter, we use a three-class classification ("COVID-19," "pneumonia," "normal") because these three automatic predictions can help doctors quickly triage patients for RT-PCR testing for COVID-19 diagnosis confirmation and choose a suitable treatment plan based on the presence and cause of infection (i.e., COVID-19 infection or non-COVID-19 infection). We investigate the effects of different data augmentation approaches on the COVID-19 CXR classification task to observe the differences between them in terms of the model's accuracy. We also explain and visualize the chosen data augmentation techniques on CXRs, (including resizing, flipping, rotating, zooming, warping, lighting, and normalizing) to understand what happens behind the scenes.

6.2 Proposed CovidXrayNet model

6.2.1 Proposed COVIDcxr dataset

COVIDcxr is the dataset of COVID-19 that we generated from two open source repositories, ChestX-Ray14 [110] and the COVID-19 Image Data Collection [117], with the associated tabular data (i.e., gender, sex, and views) for each patient. It is comprised of 960 CXR images. Our aim was to create a balanced, unbiased, and complete COVID-19 CXR dataset. We randomly selected 320 no-finding and 320 pneumonia cases from ChestX-Ray14 and 320 COVID-19 cases from the COVID-19 Image Data Collection, along with complete metadata.

There were 568 male and 392 female cases, and the average age of these subjects was about 56 years.

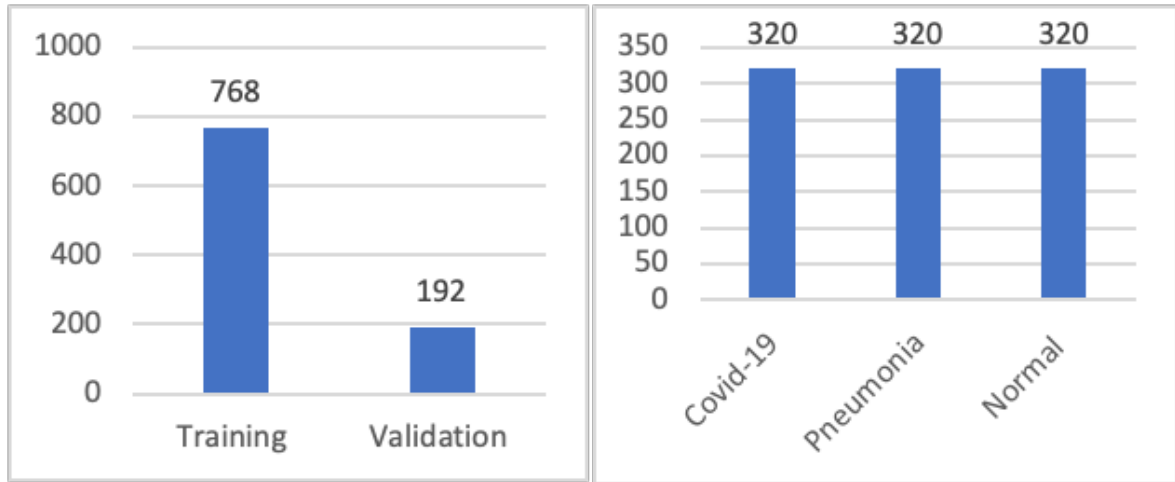
A step-by-step guide to the creation of COVIDcxr dataset is available in Appendix (B), and more details about building a single neural network based on both images (CXRs) and tabular data (sex, age, and views) can be found on <https://github.com/MaramMonshi/CovidXrayNet/tree/main/Dataset>.

6.2.2 Data preparation

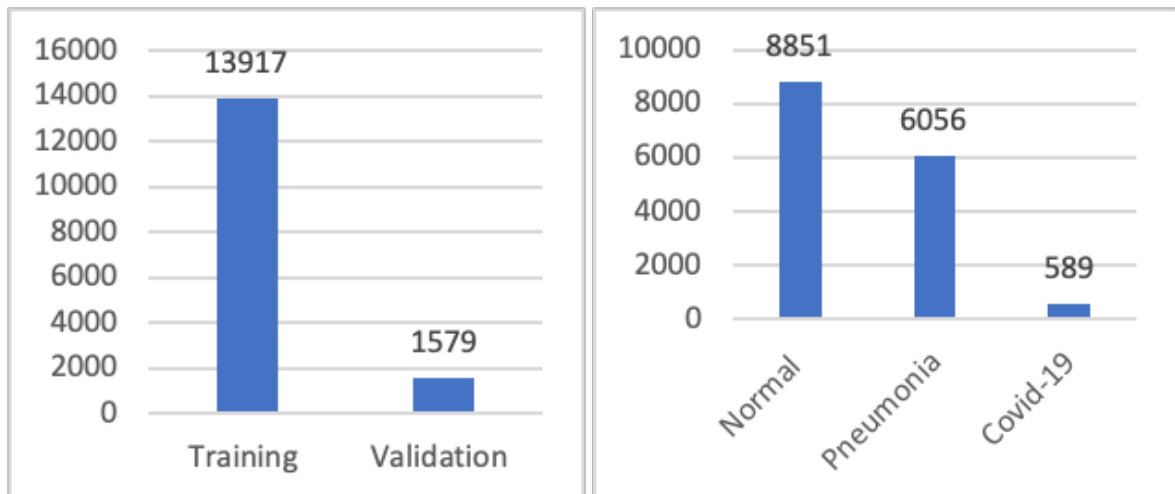
We trained CovidXrayNet on two datasets, including COVIDx [118] and our introduced COVIDcxr. Both datasets contain three classes of CXRs: COVID-19 viral infection, pneumonia (i.e., non-COVID-19 infections, such as viral and bacterial infections), and normal (i.e., no infection), as illustrated in Fig. 6.1. COVIDx is the largest and most popular dataset among researchers to date. It is expanding on a regular basis with the addition of new patient records for training while maintaining the same test dataset for consistency. We employed COVIDx-v3 in this research. However, it does not provide complete metadata for all patients and is unbalanced, as the number of cases in the COVID-19 class (589) is far lower than that in the pneumonia (6,056) or no-finding (8,851) class. COVIDcxr, however, is a balanced, unbiased, and complete COVID-19 CXR dataset.

6.2.3 Architecture

The overall structure of our proposed CovidXrayNet, which classifies a CXR as either "COVID-19," "normal," or "pneumonia," is presented in Fig. 6.2. Before feeding the CXRs to the pre-trained EfficientNet-B0 along with the optimized CNN hyperparameters, we performed several augmentation techniques on the data.



(a) COVIDx (a random split was performed [with a fixed seed] by setting 20% of the data for the validation set).



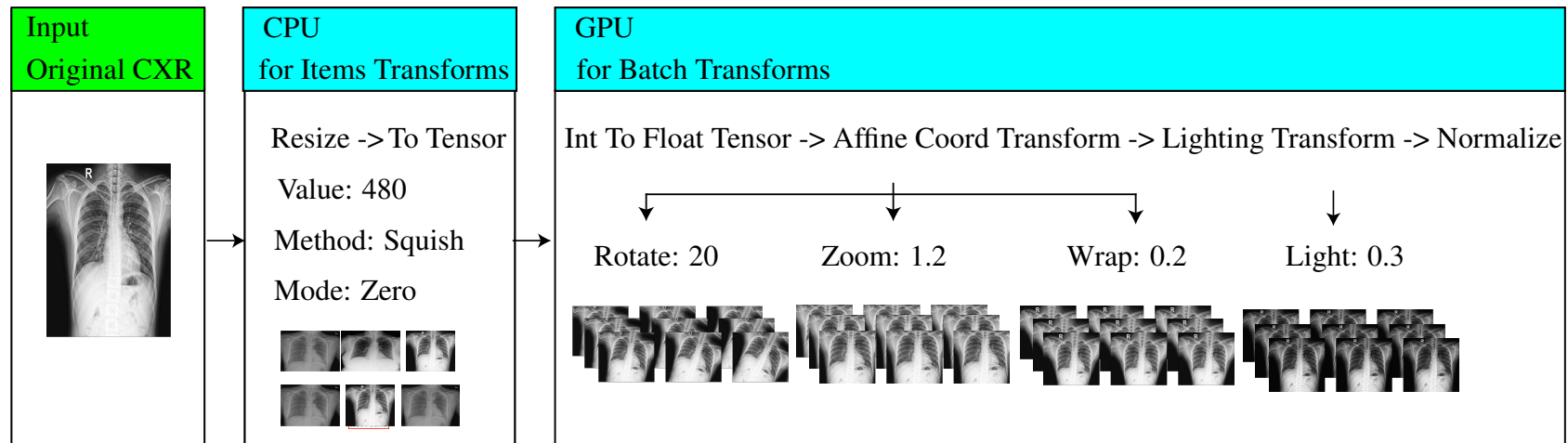
(b) COVIDx (the validation set consists of 100 COVID-19, 594 pneumonia, and 885 normal cases, in a fixed manner).

FIGURE 6.1. Dataset Distribution [119].

6.2.3.1 Data augmentation

Data augmentation enhances CNN performance [94], prevents overfitting [93], and is easy to implement [95]. Training a CNN on limited COVID-19 data inhibits its ability to generalize results to unseen data due to the overfitting issue. However, inflating a dataset using data augmentation methods adds more invariant cases and thus prevents overfitting. We defined the concept of data augmentation in Chapter 2 (Literature Review), section 2.3.4.

1. Data Augmentation Pipeline



2. Training

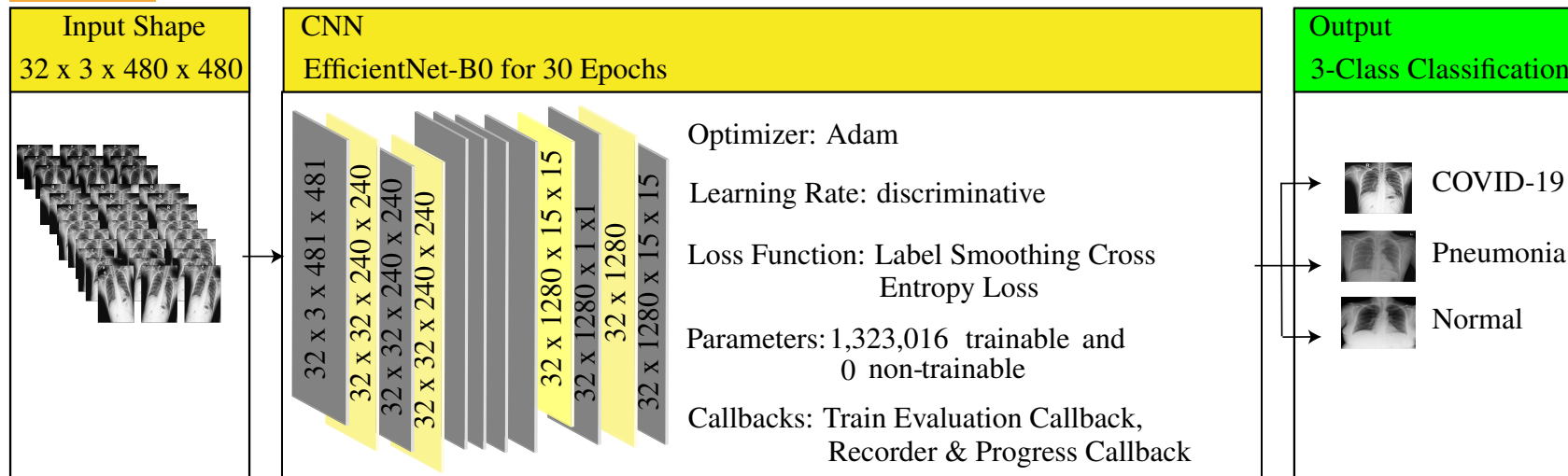


FIGURE 6.2. The CovidXrayNet Structure [119].

As there is an endless array of mappings, $\phi(x)$, we examined common data augmentation methods, including resizing, flipping, rotating, zooming, warping, lighting, and normalizing. Our investigative space was determined by consultations with practical radiologists and research on common techniques in the literature. From a radiologist's perspective, the use of portable devices that minimize the infection control issues of COVID-19 results in low-quality CXR and incorrect rotation. From a literature perspective, researchers tend to apply resizing, zooming, warping, and lighting to increase the number of cases to handle the issue of limited COVID-19 data.

First, we performed several deliberate data augmentations based on extensive experiments on COVIDcxr dataset using Residual Network (ResNet)-18 [78], as it can be seen in Table 6.4. Figure 6.3 plots all transformer techniques against each other to observe the differences between them.

At the item transformation level, we resized each CXR to 480×480 pixels by squishing the CXR on the horizontal axis on the Central Processing Unit (CPU). This constricted the ribcage toward the center while preserving all the parts of the CXR. Our method differed from the common approach in the literature, which resizes each CXR to the same aspect ratio to set the smallest dimension to a specified size and then arbitrarily crops it on the other dimension, as illustrated in Fig. 6.4. This cropping method may erase important CXR details from the edges of the image. Resizing all CXRs to a fixed size is a data augmentation prerequisite for classifying them using a CNN.

At the batch transformation level, we applied a group of optimized augmentation parameters on a Graphics Processing Unit (GPU) to minimize the number of computation and lossy operations. We used a pipeline to combine the best transformers' values. A series of experiments on the COVIDcxr dataset, with a fixed seed, was used to find the best combination of choices and orders of data augmentation that ensured ResNet-18 achieved the best accuracy, as recorded in Table 6.4.

We applied a random rotation with a maximum of 20° and 75% probability to overcome the incorrect rotation of some of the acquired images. Such low-quality CXRs are the result

TABLE 6.4. Pipeline for Data Augmentation on CXRs. For each independent parameter, we trained ResNet-18 on COVIDcxr for 30 epochs to examine the effects of various transformers on COVID-19 CXR classification.

Independent Parameter	Resize		Rotate	Zoom	Wrap	Light	Extra	(%)		
	Size	Method						Acc	AUC	F1
Resize	224*224	crop	0	0	0	0	none	78.12	90.84	78.04
		pad						79.68	90.66	79.39
		squish						74.47	88.63	74.47
	256*256	crop						79.16	92.12	79.12
		pad						76.04	90.62	75.55
		squish						78.12	90.15	78.22
	480*480	crop						80.72	94.42	80.65
		pad						82.81	94.67	82.86
	512*512	squish						83.85	94.14	83.95
crop		80.72	93.35	80.78						
pad		78.64	93.22	78.62						
Rotate	480*480	squish	0	0	0	0	none	83.85	94.14	83.95
			10					85.93	95.73	85.95
			20					86.45	96.48	86.56
			30					86.45	95.97	86.58
			50					84.89	95.72	85.03
Zoom	480*480	squish	0	1	0	0	none	83.85	94.14	83.95
				1.2				85.41	95.86	85.48
				1.3				82.29	95.77	82.37
				1.4				84.37	95.60	84.45
				1.5				81.25	95.29	81.21
Warp	480*480	squish	0	0	0	0	none	83.85	94.14	83.95
					0.1			84.37	95.36	84.42
					0.2			85.41	96.33	85.50
					0.3			84.89	96.34	84.94
Lighting	480*480	squish	0	0	0	0	none	83.85	94.14	83.95
					0.1			81.77	93.12	81.93
					0.2			83.85	94.34	83.91
					0.3			85.41	95.10	85.46
					0.4			82.81	95.34	82.97
					0.5			84.37	95.89	84.46
Flip (dihedral) Mixup (0.4) Erasing (random) Normalize (imagenet)	480*480	squish	0	0	0	0	flip	83.85	95.69	83.81
							mixup	83.33	94.88	83.29
							erase	80.72	94.11	80.91
							norm	83.85	94.14	83.95
Multiple Param (pipeline)	480*480	squish	20	1.2	0.2	0.3	flip	81.77	95.70	81.69
			20	1.2	0.2	0.3	mixup	82.81	95.86	82.48
			20	1.2	0.2	0.3	flip, norm	81.77	95.70	81.69
			20	1.2	0.2	0.3	norm	88.02	96.20	88.14

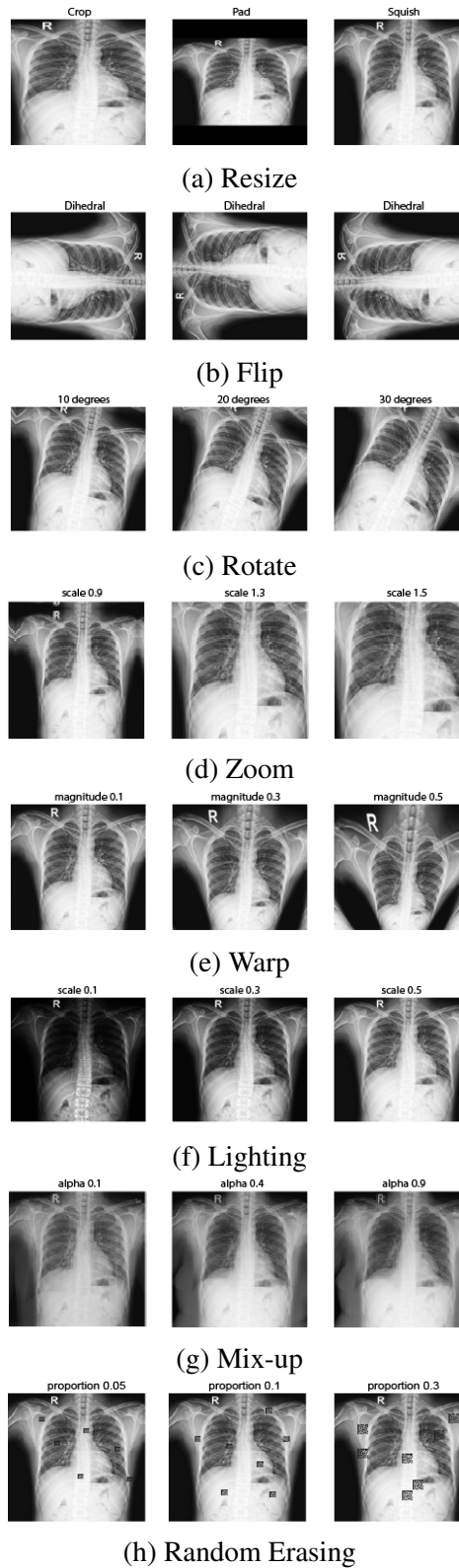
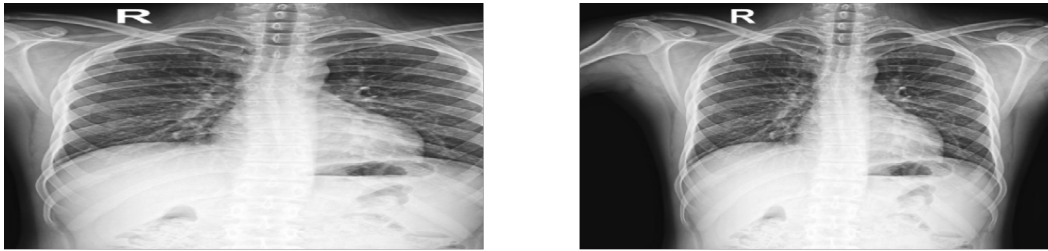


FIGURE 6.3. Visualizing Data Augmentation Effects on a CXR. The CXR is for a 25-year-old COVID-19-positive female taken from the COVID-19 Image Data Collection [119].



(a) Common approach: Crop

(b) Our method: Squish

FIGURE 6.4. Resizing Method. We propose squishing a 480×480 pixel CXR rather than cropping it to preserve important CXR details at the edges of the image [119].

of using portable devices that minimize the infection control issues of COVID-19 [235]. In addition, it is not uncommon, especially for Anteroposterior (AP) supine CXRs, for the patient to be rotated, which makes interpretation difficult. In addition to rotating CXRs, we also applied zooming, warping, and lighting as we relied on data augmentation to handle the issue of limited COVID-19 data through increasing the number of cases [94] and, hence, preventing overfitting. With a 75% probability, we zoomed the CXRs by a scale of 1.2, lightened them by a scale of 0.3, and warped them by a magnitude of 0.2. Warping and lightening augmentations may help in situations wherein patients face the X-ray device at different angles and in various lighting setups. We attempted to apply the random erasing [236] and mix-up [143] techniques, but we did not notice improved performance.

6.2.3.2 CNN architectures and hyperparameters

Next, we replaced the head of EfficientNet-B0 with a head suitable for the three-class classification and trained it for 30 epochs. To compensate for the small dataset, we performed transfer learning with the pre-trained weights from ImageNet. Then, we fine-tuned EfficientNet-B0 using one NVIDIA Tesla V100. EfficientNet scaled CovidXrayNet's width and depth according to the size of 480×480 pixels, which resulted in substantially less computational power use and fewer parameters with a high performance compared to other CNN architectures.

Table 6.5 presents the performance of the optimized data augmentation on the two datasets, COVIDcxr (small and balanced dataset) and COVIDx (large and unbalanced dataset), using the benchmark deep neural network architectures, including VGG-16, VGG-19 [76], ResNet-18, ResNet-34, ResNet-50 [78], and EfficientNet-B0 [225]. Among the various CNN architectures, EfficientNet-B0 accomplished the best results in classifying COVID-19 from COVIDcxr and COVIDx based on various evaluation metrics, such as accuracy, precision, recall, and F1 scores. Please refer to Chapter 2 (Literature Review), section 2.8.1, for more details about these evaluation metrics.

TABLE 6.5. The CNN Architectures on COVIDx and COVIDcxr. We trained the popular CNN architectures on both datasets for 30 epochs using the optimized data augmentation pipeline.

CNN	Dataset	Accuracy (%)	AUC (%)	MCC (%)	Precision (%)	Recall (%)	F1 (%)
VGG-16	COVIDcxr	80.73	94.68	72.29	82.03	81.35	80.53
VGG-19		84.90	95.67	77.74	85.31	85.26	84.92
ResNet-18		85.94	96.72	79.40	86.84	86.31	86.14
ResNet-34		79.69	94.91	70.02	80.26	80.03	79.70
ResNet-50		82.81	95.90	75.31	84.90	83.29	83.12
EfficientNet-B0		88.02	–	82.01	87.98	88.03	88.00
VGG-16	COVIDx	93.41	98.70	87.74	94.40	89.41	91.61
VGG-19		93.60	98.55	88.06	95.29	85.53	89.24
ResNet-18		93.29	98.86	87.48	95.03	86.73	90.05
ResNet-34		94.74	99.10	90.19	95.85	89.95	92.53
ResNet50		95.12	99.22	90.92	96.08	91.76	93.72
EfficientNet-B0		95.69	–	92.01	96.24	94.76	95.48

EfficientNet introduces a new and simple compound scaling technique to scale the number of layers, α , the number of channels, β , and the number of pixels, γ , in an image, representing the CNN width, depth, and resolution, respectively [225], as depicted in Eq. (6.1). This technique uses a compound coefficient, ϕ , which defines the amount of available resources to determine how to scale α , β , and γ . The constraint $(\alpha \cdot \beta^2 \cdot \gamma^2) \approx 2$ is applied in order to make sure that the total Floating-Point Operations per Second (FLOPS) do not exceed 2^ϕ .

$$\begin{aligned}
&\text{depth: } d = \alpha^\phi \\
&\text{width: } w = \beta^\phi \\
&\text{resolution: } r = \gamma^\phi \\
&\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
&\alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned} \tag{6.1}$$

CovidXrayNet was based on baseline network EfficientNet-B0, where the optimal values are $\alpha = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$. Using this multi-objective neural architecture search, we optimized both the accuracy and FLOPS. Although the original EfficientNet-B0 uses the standard input size of 224×224 pixels, it handles 480×480 CXR pixels perfectly. We customized the final output layer of EfficientNet-B0 to suitably perform multi-class classification.

In addition, CovidXrayNet is coupled with a narrow-wide-narrow structure [237], squeeze-and-excitation blocks [81], and swish activation function [238]. In the narrow-wide-narrow structure, the residual blocks are coupled with 3×3 and 5×5 depth-wise convolution. The squeeze-and-excitation blocks allowed CovidXrayNet to alter the weighting of each feature map adaptively. The swish activation function, defined by $\varphi(x) := \frac{x}{1+e^{-\beta x}}$, is a smooth function that interpolates non-linearly between a linear function for $\beta = 0$ and the rectified linear unit function for $\beta \rightarrow \infty$. During back-propagation, the swish activation function mitigated the issue of vanishing gradient.

Furthermore, we studied various CNN hyperparameters on COVIDcxr and COVIDx, including the loss function, the number of epochs, and the batch size, as demonstrated in Table 6.6 and Table 6.7. Based on this trial-and-error method, we selected the optimal hyperparameters for EfficientNet-B0 on the COVIDx dataset including the label smoothing [239] of the cross-entropy loss function, 30 epochs, and a batch size of 32. The label smoothing for our three-class problem is presented in Eq. (6.2), where $(1 - \epsilon)$ is the prediction of the correct class and ϵ is the prediction of the other two classes. In this formula, $ce(x)$ donates the standard cross-entropy loss of x , ϵ is a small positive number, i is the correct class, and N is

the number of classes. This regularization technique improved CovidXrayNet’s performance and robustness by computing the cross-entropy with a weighted mixture of the hard targets from the COVIDx dataset using the uniform distribution.

$$\text{cross entropy loss} = (1 - \epsilon) ce(i) + \epsilon \sum \frac{ce(j)}{N} \quad (6.2)$$

We fine-tuned CovidXrayNet using fit-one-cycle policy [205] and discriminative learning rates [195]. Equation (6.3) defines this discriminative fine-tuning technique, where CovidXrayNet’s parameters θ are split into $\{\theta^1, \dots, \theta^L\}$, and the learning rates, η , are split into $\{\eta^1, \dots, \eta^L\}$ at time step "t" for the number of layers, "L". Using this function, we started with a learning rate of $2e - 3$ and then automatically adjusted this value for both COVIDx and COVIDcxr, where the gradient of the CovidXrayNet’s objective function is $\nabla_{\theta^l} j$.

$$\theta_t^l = \theta_{t-l}^l - \eta^l \times \nabla_{\theta^l} j(\theta) \quad (6.3)$$

6.3 Experiment

We used the PyTorch software [103], the fastai library [105], an n1-highmem-8 (8 vCPUs, 52 GB memory) machine, and one NVIDIA Tesla V100 GPU. Fastai is a DL library that enables the implementation of CovidXrayNet with its unique ability to join several transformers inside a pipeline that manages the minimum number of computations and lossy operations.

Our proposed CovidXrayNet is available in Appendix (B), and more details about the implementation of this model can be found on <https://github.com/MaramMonshi/CovidXrayNet>.

TABLE 6.6. Optimizing CNN Hyperparameters using COVIDcxr. For each independent parameter, we trained several architectures on COVIDcxr to examine the effects of various hyperparameters on the accuracy of COVID-19 CXR classification.

CNN	Epoch	Batch Size	Loss Function	Acc (%)	MCC (%)	F1 (%)
VGG-16	10	32	Cross Entropy	77.08	68.15	76.26
	20	32	Cross Entropy	77.60	66.71	77.43
	30	32	Cross Entropy	80.73	72.29	80.53
	40	32	Cross Entropy	83.33	75.51	83.31
	30	8	Cross Entropy	85.42	79.20	85.43
	30	16	Cross Entropy	84.38	76.61	84.27
	30	32	Label Smoothing	79.17	69.62	78.91
VGG-19	10	32	Cross Entropy	78.65	68.38	78.89
	20	32	Cross Entropy	82.81	74.25	82.96
	30	32	Cross Entropy	84.90	77.74	84.92
	40	32	Cross Entropy	84.38	76.66	84.35
	30	8	Cross Entropy	84.90	78.36	84.96
	30	16	Cross Entropy	82.81	74.90	82.74
	30	32	Label Smoothing	85.42	78.30	85.51
ResNet-18	10	32	Cross Entropy	81.25	73.69	81.25
	20	32	Cross Entropy	82.29	74.21	82.45
	30	32	Cross Entropy	85.94	79.40	86.14
	40	32	Cross Entropy	85.42	78.16	85.37
	30	8	Cross Entropy	81.25	73.56	81.39
	30	16	Cross Entropy	82.29	74.20	82.37
	30	32	Label Smoothing	84.38	76.95	84.46
ResNet-34	10	32	Cross Entropy	81.25	72.10	81.20
	20	32	Cross Entropy	81.25	71.93	80.91
	30	32	Cross Entropy	79.69	70.02	79.70
	40	32	Cross Entropy	81.25	71.94	81.23
	30	8	Cross Entropy	86.46	80.12	86.54
	30	16	Cross Entropy	85.94	79.00	85.87
	30	32	Label Smoothing	83.85	76.08	83.85
ResNet-50	10	32	Cross Entropy	81.77	73.18	82.09
	20	32	Cross Entropy	84.90	77.32	84.93
	30	32	Cross Entropy	82.81	75.31	83.12
	40	32	Cross Entropy	85.42	78.12	85.45
	30	8	Cross Entropy	86.46	80.49	86.52
	30	16	Cross Entropy	86.98	80.84	87.16
	30	32	Label Smoothing	83.85	76.21	84.05
EfficientNet-B0	10	32	Cross Entropy	83.33	75.36	83.65
	20	32	Cross Entropy	84.38	76.67	84.41
	30	32	Cross Entropy	88.02	82.01	88.00
	40	32	Cross Entropy	85.42	78.10	85.42
	30	8	Cross Entropy	88.02	82.06	87.89
	30	16	Cross Entropy	86.98	80.45	86.99
	30	32	Label Smoothing	88.54	82.83	88.62

TABLE 6.7. Optimizing CNN Hyperparameters using COVIDx. For each independent parameter, we trained several architectures on COVIDx to examine the effects of various hyperparameters on the accuracy of COVID-19 CXR classification.

CNN	Epoch	Batch Size	Loss Function	Acc (%)	MCC (%)	F1 (%)
VGG-16	10	32	Cross Entropy	92.08	85.20	86.99
	20	32	Cross Entropy	93.35	87.56	90.10
	30	32	Cross Entropy	93.41	87.74	91.61
	40	32	Cross Entropy	94.24	89.25	91.99
	30	8	Cross Entropy	93.86	88.56	91.03
	30	16	Cross Entropy	94.30	89.38	92.00
	30	32	Label Smoothing	94.05	88.88	91.35
VGG-19	10	32	Cross Entropy	92.53	86.04	87.29
	20	32	Cross Entropy	93.98	88.77	91.57
	30	32	Cross Entropy	93.60	88.06	89.24
	40	32	Cross Entropy	93.29	87.46	88.72
	30	8	Cross Entropy	94.49	89.73	92.14
	30	16	Cross Entropy	94.93	90.56	92.79
	30	32	Label Smoothing	93.79	88.40	90.10
ResNet-18	10	32	Cross Entropy	93.10	87.08	88.43
	20	32	Cross Entropy	93.60	88.06	90.07
	30	32	Cross Entropy	93.29	87.48	90.05
	40	32	Cross Entropy	93.86	88.53	90.87
	30	8	Cross Entropy	94.17	89.11	91.17
	30	16	Cross Entropy	94.43	89.60	92.49
	30	32	Label Smoothing	94.30	89.35	91.58
ResNet-34	10	32	Cross Entropy	94.05	88.89	91.41
	20	32	Cross Entropy	94.62	89.97	93.32
	30	32	Cross Entropy	94.74	90.19	92.53
	40	32	Cross Entropy	94.43	89.63	93.38
	30	8	Cross Entropy	94.87	90.44	92.43
	30	16	Cross Entropy	95.31	91.28	94.50
	30	32	Label Smoothing	94.62	89.96	92.50
ResNet-50	10	32	Cross Entropy	94.93	90.55	92.62
	20	32	Cross Entropy	94.81	90.34	93.37
	30	32	Cross Entropy	95.12	90.92	93.72
	40	32	Cross Entropy	94.81	90.35	93.14
	30	8	Cross Entropy	93.03	87.01	91.99
	30	16	Cross Entropy	95.57	91.76	95.35
	30	32	Label Smoothing	95.12	90.91	93.36
EfficientNet-B0	10	32	Cross Entropy	95.69	91.99	94.52
	20	32	Cross Entropy	95.19	91.02	93.38
	30	32	Cross Entropy	95.69	92.01	95.48
	40	32	Cross Entropy	95.00	90.72	95.00
	30	8	Cross Entropy	94.68	90.16	93.25
	30	16	Cross Entropy	95.38	91.40	94.88
	30	32	Label Smoothing	95.82	92.24	96.16

6.4 Results and discussion

6.4.1 Quantitative evaluation

We computed the accuracy, macro average precision, macro average recall, macro F1 score, Area Under the Receiver Operating Characteristic Curve (AUC) [180], and Matthews Correlation Coefficient (MCC) [181] of CovidXrayNet in distinguishing between the three classes ("COVID-19," "pneumonia," and "normal"). We used this combination of evaluation metrics to be the criterion for selecting the best model to avoid an inaccurate conclusion. For example, since accuracy depends mostly on the number of samples in each class, CNN-based models perform seemingly well in imbalanced datasets, such as COVIDx. In addition, we followed the macro approach because it considers all classes as basic elements of the calculation [182] (i.e., each class has the same weight in the average regardless of its size). We explained these evaluation metrics in Chapter 2 (Literature Review), section 2.8.1.

In order to evaluate our proposed data augmentation pipeline, we compared the reported results of VGG-19 and ResNet-50 in the COVID-Net paper [118] with our results on the COVIDx dataset, as recorded in Table 6.8. With only 30 epochs of learning cycles, the accuracy of VGG-19 increased by 11.93%, while the accuracy of ResNet-50 improved by 4.97%. The results clearly indicated the effect of our proposed method on enhancing the accuracy of COVID-19 classification from CXRs.

TABLE 6.8. Comparing our Optimized Data Augmentation Pipeline and CNN Hyperparameters with the Benchmark. Both papers used VGG-19 and ResNet-50 on the COVIDx dataset but with different transformers and hyperparameters.

CNN	Paper	Parameters (M)	Accuracy (%)	AUC (%)	MCC (%)	F1 (%)
VGG-19	COVID-Net [118]	20	83.00	–	–	–
	CovidXrayNet		94.93	98.69	90.56	92.79
ResNet-50	COVID-Net [118]	25	90.60	–	–	–
	CovidXrayNet		95.57	99.29	91.76	95.35

Table 6.9 compares CovidXrayNet to other studies in the literature that are based on a three-class classification. We achieved better accuracy (95.82%) over the other models, including

DarkCovidNet (87.02%), COVID-Net (93.30%), and MobileNet v2 (93.48%). Furthermore, the F1 score for CovidXrayNet (96.16%) was higher than that of DarkCovidNet (87.37%), and the precision score of CovidXrayNet (96.93%) was better than that of DarkCovidNet (89.96%). Significantly, the overall sensitivity of CovidXrayNet was 95.43%. Our reported results are reproducible. We used the same test dataset as COVID-Net.

TABLE 6.9. Comparing CovidXrayNet with the Benchmark. All models were based on a three-class COVID-19 classification; COVID-Net and CovidXrayNet employed the COVIDx dataset.

Model	Accuracy (%)	MCC (%)	Precision (%)	Recall (%)	F1 (%)
DarkCovidNet [151]	87.02	–	89.96	–	87.37
COVID-Net [118]	93.30	–	–	–	–
MobileNet v2 [154]	93.48	–	–	–	–
CovidXrayNet	95.82	92.24	96.93	95.43	96.16

6.4.2 Qualitative evaluation

We ensured the robustness of CovidXrayNet by sharing its top prediction errors and actual labels with expert radiologists (refer to Fig. 6.5). CovidXrayNet classified four patients with COVID-19 as having pneumonia. Since COVID-19 is a subset of pneumonia diseases, the diagnosis was correct, but the interpretation was not. For this reason, CovidXrayNet could only offer a second opinion to the radiologist in the clinical setting.

6.4.3 Optimization in deep learning

We aimed to implement an AI model, CovidXrayNet, that could identify COVID-19 infection based on CXRs. CovidXrayNet optimizes data augmentation to enable CNN models to observe visual features that are not noticeable to a radiologist’s eye. With data augmentation, CNN models will generalize better results. However, the implications of choosing efficient and effective augmentation techniques depend on the dataset at hand. Using CXRs with COVID-19 datasets, we performed a separate search phase that was computationally expensive. Recently developed methods, such as RandAugment [240] and AutoAugment

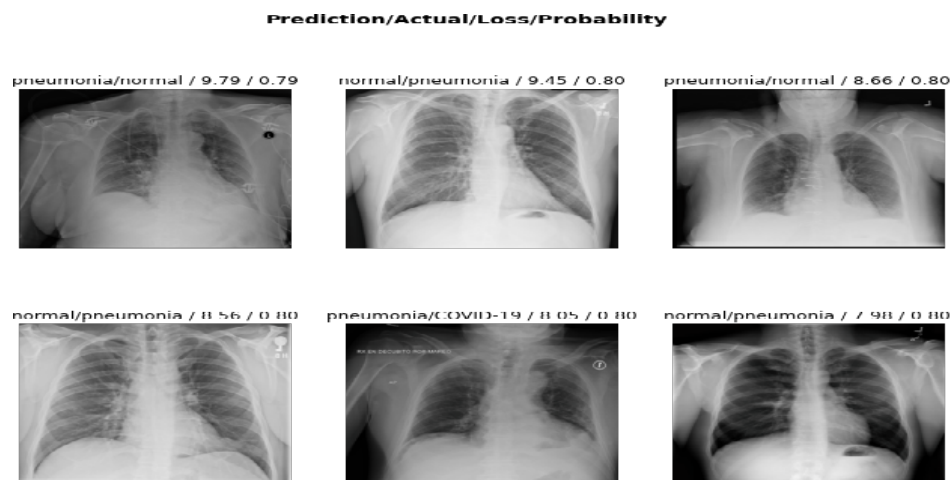


FIGURE 6.5. Top Prediction Errors Generated by CovidXrayNet on COVIDx Test Dataset [119].

[241], suggest removing the need for a search phase to reduce the parameter space for data augmentation. However, incorrect choices in the COVID-19 classification task may lead to the erasure or dilution of vital features.

Notably, the individual data augmentation methods yielded a minor increased task performance, as seen in Table 6.4. For example, the optimal warping value improved the classification task accuracy by only 1.56%. However, a combination of these optimized methods (i.e., our proposed data augmentation pipeline and CNN hyperparameters) increased the performance significantly, as can be seen in Table 6.8. It increased the accuracy of the popular CNN architectures VGG-19 and ResNet-50, by 11.93% and 4.97%, respectively.

We found that EfficientNet-B0 performed well for COVID-19 CXR classification with the following data augmentation pipeline: squishing the CXR to 480×480 pixels, rotating it by 20° , zooming it by 1.2 scale, warping it by 0.2 magnitude, lighting it by 0.3 scale, and normalizing it. Additionally, the label-smoothing cross-entropy loss function, with a batch size of 32 with 30 epochs, increased the accuracy of CovidXrayNet on the COVIDx dataset. EfficientNet is rapidly becoming the DL practitioners' choice over ResNet for many

classification tasks. It allows practitioners to use the minimum FLOPS while achieving the best possible accuracy by compound scaling the network's depth, width, and input resolution.

6.4.4 Limitation and future direction

While CovidXrayNet performed well as a whole (see Fig. 6.6), it misidentified four patients with COVID-19 as having pneumonia, and one patient with COVID-19 as being normal (refer to the confusion matrix in Fig. 6.7). However, it is important to limit the number of missed patients with COVID-19 to be isolated as well as the number of patients with false-positive COVID-19 to avoid an unnecessary burden on clinical sites. Therefore, CovidXrayNet is still in the research stage and is not suitable for direct clinical diagnosis. It could be built upon and optimized with additional data augmentation and better CNN hyperparameters.

Without conducting a proper clinical study, the achieved accuracy of CovidXrayNet (95.82%) on the COVIDx dataset did not indicate that CovidXrayNet is sufficient for detecting COVID-19 from CXRs. Our aim is to empower this research wave through our optimized data augmentation pipeline and CNN hyperparameters. Therefore, we have released the source code of CovidXrayNet to enable researchers to reproduce the results and experiment on different datasets.

As there is an endless array of transformations, our work evaluated common augmentation techniques in the CXR classification literature (i.e., resizing values, resizing methods, rotating, zooming, warping, lighting, flipping and normalizing), recently proposed methods (i.e., mix-up and random erasing), and combinations of these methods. Future research could enhance our model with de-noising or segmentation steps. In addition, the proposed data augmentation pipeline was tested only on a three-class classification task ("COVID-19," "normal," or "pneumonia"). Researchers may investigate the effects of the proposed technique on the detection of other common CXR observations including atelectasis, cardiomegaly, consolidation, edema, enlarged cardiomeastinum, fracture, lung lesion, lung opacity, pleural effusion, pleural other, pneumonia, and pneumothorax.

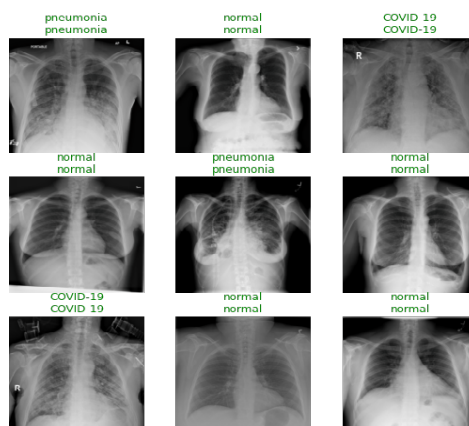


FIGURE 6.6. Randomly Generated Results for CovidXrayNet on COVIDx Test Dataset [119].

Confusion matrix

	Actual	COVID-19	normal	pneumonia	
	COVID-19	95	1	4	
	normal	0	857	28	
	pneumonia	0	33	561	
		Predicted	COVID-19	normal	pneumonia

FIGURE 6.7. Confusion Matrix for CovidXrayNet on COVIDx Test Dataset [119].

Designing a fair testing protocol can be highly challenging. Datasets with large differences have been merged in order to respond to the global challenge of quickly identifying COVID-19 [117]. The COVIDx and COVIDcxr datasets were collected from public sources. They were also indirectly collected from hospitals and physicians. For COVIDx, we tested our model with the official split recommended by the COVIDx paper to allow for future comparison. For the COVIDcxr dataset, we released the dataset generation scripts. Future research should assess the validity of the available testing protocol by validating the COVID-19 CXRs with clinical experts and determining the ground truth.

```
mixed_dl.show_batch()
```

	sex	view	age	finding
0	M	PA	41.000000	Pneumonia
1	F	AP	67.000000	No Finding
2	M	PA	75.000000	COVID-19
3	M	PA	21.000001	Pneumonia
4	M	PA	79.000000	No Finding
5	M	PA	68.000000	No Finding
6	M	PA	75.000000	No Finding
7	M	AP Supine	21.000001	COVID-19

FIGURE 6.8. Data Loader from COVIDcxr that Combines both Tabular Data and CXRs [119].

COVIDcxr is suitable for building a single neural network based on both images (CXRs) and tabular data (sex, age, and views), as can be seen in Fig. 6.8. However, we did not observe better performance for such a model than for a linear model with embedding. Even though a multi-modal network, with multiple input modalities, receives more information, it is often prone to overfitting [242]. Future research may explore training multi-modal classification networks based on the COVIDcxr dataset using various CNN architectures and hyperparameters.

6.5 Summary and conclusion

The rapid spread of the COVID-19 pandemic along with the limited number of RT-PCR test kits and qualified radiologists, has led to the need for accurate automated detection systems.

Chest radiography is one of the main imaging methods that are fast, non-invasive, affordable, and possibly able to be completed at the bedside to monitor the progression of COVID-19 infection. However, radiologists with expertise in CXR interpretation may not be available at every institution.

In this chapter, we proposed CovidXrayNet based on EfficientNet-B0 and our optimization results. We evaluated CovidXrayNet on two datasets, including our generated balanced COVID_{cxr} dataset (960 CXRs) and the benchmark COVID_x dataset (15,496 CXRs). With only 30 epochs of training, CovidXrayNet achieved a state-of-the-art accuracy of 95.82% on the COVID_x dataset in the three-class classification task (COVID-19, normal, or pneumonia).

We have demonstrated that optimizing data augmentation and CNN hyperparameters results in outstanding effects on the automatic extraction of features from CXRs related to the diagnosis of COVID-19. CovidXrayNet only required 30 learning cycles to process a CXR yet achieved 95.82% accuracy on the COVID_x dataset.

Conclusion

Throughout this dissertation, we made contributions to the five areas of Deep Learning (DL) in Chest X-ray (CXR): the understanding of CXR and DL intersections (Chapter 2), report labeling (Chapter 3), binary CXR classification (Chapter 4), multi-label CXR classification (Chapter 5), and multi-class CXR classification of Coronavirus Disease 2019 (COVID-19) (Chapter 6). In this final chapter, we summarize the proposed methods (section 7.1), recapitulate our findings (section 7.2), and present an outlook on future research (section 7.3).

7.1 Summary of the proposed methods

In this thesis, we investigated DL in CXRs, from report labeling to image classification. We first extensively studied the literature in Chapter 2 to provide practical guidelines for this relatively recent direction of research. We explained radiology text and image structures, DL algorithms, available datasets, report labeling, image classification, report generation, and model evaluation methods.

Then, in Chapter 3, we contributed to the understanding of one of the main challenges in CXR interpretation, which is extracting multiple labels from reports. We proposed CXRlabeler, a novel framework that combines the strengths of both Language Model (LM) fine-tuning and classifier fine-tuning to achieve highly accurate automated CXR report labeling.

Lastly, we proposed three novel Convolutional Neural Network (CNN) classifiers: MultiView-Model in Chapter 4, Xclassifier in Chapter 5, and CovidXrayNet in Chapter 6. MultiView-Model tackled the binary classification task through a stage-wise training technique that used

frontal and lateral CXRs. Xclassifier was a multi-label classifier that used distributed DL methods and an anti-aliasing filter. It reduced the computational complexity while preserving the classifier accuracy. For the multi-class classification problem, we investigated the role of data augmentation and CNN hyperparameters in increasing the accuracy of detecting COVID-19. Based on this investigation, we proposed CovidXrayNet, which achieved high accuracy on two datasets: the benchmark dataset and our introduced COVIDcxr dataset.

7.2 Summary of findings

Over the course of this thesis, we strove to address the research questions we laid out initially in Chapter 1, aiming to build efficient and accurate CXR classifiers. We presented multiple novel methods for different classification scenarios and evaluated them across multiple datasets. We now summarize our contributions and findings.

Understanding the gaps in the literature (RQ 1.): We extensively reviewed the existing literature on the DL path employed in radiology, from report labeling to image classification and image captioning. We noticed that binary, multi-class, and multi-label classifications are the most common classification problems in radiology. However, limited existing methods specifically consider efficiency in their proposed frameworks, such as by optimizing data augmentation, CNN hyperparameters, CXR formats, and training techniques.

Extracting multiple labels from CXR reports (RQ 2.): We realized that DL models can be beneficial in extracting labels from CXR reports, even in the absence of trustworthy training labels. Using the benefits of both LM fine-tuning and classifier fine-tuning can result in highly accurate automated CXR report labeling. Our tests on different datasets showed that this method of sequential transfer learning works well in a different language environment.

Classifying CXRs using DL techniques (RQ 3.): We found that combining multiple views of CXRs in a training set can result in better binary classification. In addition, reducing the training time and increasing the accuracy can be achieved by observing the model's performance in a stage-wise training manner. Additionally, we demonstrated the benefit

of combining several recent DL approaches with the CXR classifiers, including transfer learning, fine-tuning, fit-one-cycle functions, and discriminative learning rates. Among the various CNN architectures, Residual Network (ResNet), Densely Connected Convolutional Network (DenseNet), and EfficientNet are the best architectures for the CXR classification problem.

Building efficient multi-label classifiers (RQ 4.): We noticed that the Distributed Data Parallel (DDP) technique is a true data parallelism process. It is useful for performing multi-processes on devices of just a single machine to accomplish ideal memory utilization and Graphics Processing Unit (GPU) computation for multi-label CXR classification. Deep learning models that use CXRs in the Joint Photographic Experts Group (JPEG) format are faster and more accurate than the ones that use the Digital Imaging and Communications in Medicine (DICOM) format for detecting multiple diseases simultaneously. Anti-aliasing filters can be used to increase the accuracy of CXR classifiers due to the shift variant nature of CNNs.

Optimizing CNN classifiers (RQ 5.): We discovered that optimizing data augmentation and CNN hyperparameters resulted in outstanding effects on the automatic extraction of features from CXRs related to the diagnosis of COVID-19. An optimal data augmentation pipeline for CXR classification may include squishing the CXR to 480×480 pixels, rotating it by 20° , zooming in by 1.2 scale, warping it by 0.2 magnitude, lighting it by 0.3 scale, and normalizing it. Our CovidXrayNet only required 30 learning cycles to process a CXR but achieved 95.82% accuracy on the COVIDx dataset. Our optimization increased the accuracy of the popular CNN architectures such as the Visual Geometry Group VGG-19 and ResNet-50 by 11.93% and 4.97%, respectively.

7.3 Future work

The number of contributions in the field of CXR DL research has been growing due to the release of extremely valuable public datasets in recent years, with more than 739,000 labeled CXRs released between 2017 and 2022 [173]. However, there are multiple technical and

clinical issues to be considered in relation to their use, including label and image quality, CXR captioning, explainable systems, and clinical transition.

Label and image qualities: Public datasets are labeled through one or more of the following methods: report parsing, DL, radiologist interpretations of reports, radiologist interpretations of CXRs, radiologist cohort agreement on CXRs, and laboratory tests. For evaluation and comparison of the state-of-the-art models, it is recommended to use "gold standard" test data labels, such as those acquired by radiologist cohort agreement on CXRs. Image quality is another factor that is often overlooked by the CXR research community, whose datasets often have reduced-quality images. Although Chapter 5 studies the effect of image quality on valuable diagnostic information, more research should address this issue.

CXR captioning: Beyond CXR classification and single-sentence-based descriptions, generating coherent radiology paragraphs has recently attracted researchers. This presents a more practical and challenging application that can bridge visual medical features with radiologist interpretations. Notably, CNN and Recurrent Neural Network (RNN) have quickly become popular choices for mining radiology images and text, respectively. The main problem now is obtaining ImageNet-level captions on an extensive collection of medical images.

Explainable systems: The term "explainable Artificial Intelligence (AI)" is frequently used to describe an ongoing challenge for DL researchers to tackle. In DL for CXRs, most past research has focused on image-level predictions to generate classification labels without logical evidence, which presents trust issues for radiologists. Our MultiViewModel in Chapter 4 [146] and Mitra et al. [243] used heatmaps, such as gradient-weighted class activation maps [244] and saliency maps [245], to indicate which regions in an image are relevant to the results. However, the lack of a sound assessment of their accuracy is a concern. Future research should have a localization function that shows where the abnormalities in a CXR are located (e.g., identification of nodule location with a bounding box). Additionally, many conditions, such as emphysema, which is indicated by irregular radiolucency throughout the lung, may be difficult to explain using a heatmap. In these cases, an image could be labeled (e.g., positive or negative) for a known sequence of radiological aspects related to the

condition being diagnosed, or segmentation information could be used in the classification. This is one way to build clinically explainable systems.

Clinical transition: Due to legal and ethical policies as well as technical hurdles, the transition of DL CXR research to clinical use is uncommon [246]. Despite the fact that AI research for thoracic radiology is among the most advanced in radiology, its use in clinical practice is still limited. Leeuwen et al. [247] estimated that more than 40 AI applications have been marked by the European Conformity Marking for thoracic radiology, where half of the products are designed for chest radiography analysis and the other half for chest Computed Tomography (CT). After neuroradiology, this is the specialization in radiology with the most AI applications. Researchers who want to create therapeutically relevant technologies should pay greater attention to the workflow and requirements of radiologists or clinicians. As a Two-Dimensional (2D) scan, a CXR can be examined by a radiologist relatively quickly. Therefore, the goal for DL researchers is to develop systems that save radiologists time, prioritize critical cases, or increase the sensitivity/specificity of their results.

Bibliography

- [1] *Statistics » Diagnostic Imaging Dataset*. [Online]. Available: <https://www.england.nhs.uk/statistics/statistical-work-areas/diagnostic-imaging-%20dataset/> (visited on 23/05/2021).
- [2] G. S. Lodwick, T. E. Keats and J. P. Dorst, ‘The coding of roentgen images for computer analysis as applied to lung cancer,’ *Radiology*, vol. 81, no. 2, pp. 185–200, 1963, ISSN: 0033-8419.
- [3] H. C. Becker, W. J. Nettleton, P. H. Meyers, J. W. Sweeney and C. M. Nice, ‘Digital computer determination of a medical diagnostic index directly from chest X-ray images,’ *IEEE Transactions on Biomedical Engineering*, no. 3, pp. 67–72, 1964, ISSN: 0018-9294.
- [4] P. H. Meyers, C. M. Nice Jr, H. C. Becker, W. J. Nettleton Jr, J. W. Sweeney and G. R. Meckstroth, ‘Automated computer analysis of radiographic images,’ *Radiology*, vol. 83, no. 6, pp. 1029–1034, 1964, ISSN: 0033-8419.
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ‘Imagenet: A large-scale hierarchical image database,’ in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255, ISBN: 1424439922.
- [6] I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
- [7] X. Glorot, A. Bordes and Y. Bengio, ‘Deep sparse rectifier neural networks,’ in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
- [8] M. Z. Alom, T. M. Taha, C. Yakopcic *et al.*, ‘The history began from alexnet: A comprehensive survey on deep learning approaches,’ *arXiv preprint arXiv:1803.01164*, 2018.

- [9] S. Pouyanfar, S. Sadiq, Y. Yan *et al.*, ‘A survey on deep learning: Algorithms, techniques, and applications,’ *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018, ISSN: 0360-0300.
- [10] B. Shickel, P. J. Tighe, A. Bihorac and P. Rashidi, ‘Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis,’ *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1589–1604, 2017, ISSN: 2168-2194.
- [11] O. Ruuskanen, E. Lahti, L. C. Jennings and D. R. Murdoch, ‘Viral pneumonia,’ *The Lancet*, vol. 377, no. 9773, pp. 1264–1275, 2011.
- [12] *WHO Coronavirus (COVID-19) Dashboard | WHO Coronavirus Disease (COVID-19) Dashboard*. [Online]. Available: <https://covid19.who.int/> (visited on 31/03/2021).
- [13] J. Irvin, P. Rajpurkar, M. Ko *et al.*, ‘Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison,’ in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 590–597.
- [14] J. Ker, L. Wang, J. Rao and T. Lim, ‘Deep learning applications in medical image analysis,’ *Ieee Access*, vol. 6, pp. 9375–9389, 2017, ISSN: 2169-3536.
- [15] B. Van Ginneken, C. M. Schaefer-Prokop and M. Prokop, ‘Computer-aided diagnosis: how to move from the laboratory to the clinic,’ *Radiology*, vol. 261, no. 3, pp. 719–732, 2011, ISSN: 0033-8419.
- [16] M. Kohli, L. M. Prevedello, R. W. Filice and J. R. Geis, ‘Implementing machine learning in radiology practice and research,’ *American journal of roentgenology*, vol. 208, no. 4, pp. 754–760, 2017, ISSN: 0361-803X.
- [17] J. Wang, X. Yang, H. Cai, W. Tan, C. Jin and L. Li, ‘Discrimination of breast cancer with microcalcifications on mammography by deep learning,’ *Scientific reports*, vol. 6, no. 1, pp. 1–9, 2016, ISSN: 2045-2322.
- [18] J.-Z. Cheng, D. Ni, Y.-H. Chou *et al.*, ‘Computer-aided diagnosis with deep learning architecture: applications to breast lesions in US images and pulmonary nodules in CT scans,’ *Scientific reports*, vol. 6, no. 1, pp. 1–13, 2016, ISSN: 2045-2322.

- [19] M. P. McBee, O. A. Awan, A. T. Colucci *et al.*, ‘Deep learning in radiology,’ *Academic radiology*, vol. 25, no. 11, pp. 1472–1480, 2018, ISSN: 1076-6332.
- [20] A. Esteva, A. Robicquet, B. Ramsundar *et al.*, ‘A guide to deep learning in healthcare,’ *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019, ISSN: 1546-170X.
- [21] D. Ravì, C. Wong, F. Deligianni *et al.*, ‘Deep learning for health informatics,’ *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016, ISSN: 2168-2194.
- [22] G. Litjens, T. Kooi, B. E. Bejnordi *et al.*, ‘A survey on deep learning in medical image analysis,’ *Medical image analysis*, vol. 42, pp. 60–88, 2017, ISSN: 1361-8415.
- [23] F. Wang, L. P. Casalino and D. Khullar, ‘Deep learning in medicine—promise, progress, and challenges,’ *JAMA internal medicine*, vol. 179, no. 3, pp. 293–294, 2019, ISSN: 2168-6106.
- [24] A. Akay and H. Hess, ‘Deep learning: current and emerging applications in medicine and technology,’ *IEEE journal of biomedical and health informatics*, vol. 23, no. 3, pp. 906–920, 2019, ISSN: 2168-2194.
- [25] *Imaging and radiology: MedlinePlus Medical Encyclopedia*. [Online]. Available: <https://medlineplus.gov/ency/article/007451.htm> (visited on 30/04/2021).
- [26] H.-C. Shin, K. Roberts, L. Lu, D. Demner-Fushman, J. Yao and R. M. Summers, ‘Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2497–2506.
- [27] C. Y. Li, Z. Hu, X. Liang and E. P. Xing, ‘Hybrid retrieval-generation reinforced agent for medical image report generation,’ *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 1530–1540, 2018, ISSN: 10495258.
- [28] M. M. A. Monshi, J. Poon and V. Chung, ‘Deep learning in generating radiology reports: A survey,’ *Artificial Intelligence in Medicine*, p. 101 878, 2020, ISSN: 0933-3657.

- [29] P. L. Schuyler, W. T. Hole, M. S. Tuttle and D. D. Sherertz, 'The UMLS Metathesaurus: representing different views of biomedical concepts.,' *Bulletin of the Medical Library Association*, vol. 81, no. 2, p. 217, 1993.
- [30] C. P. Langlotz, *Radlex: A new method for indexing online educational materials*, 2006.
- [31] D. Demner-Fushman, M. D. Kohli, M. B. Rosenman *et al.*, 'Preparing a collection of radiology examinations for distribution and retrieval,' *Journal of the American Medical Informatics Association*, vol. 23, no. 2, pp. 304–310, 2016, ISSN: 1527-974X.
- [32] B. Jing, P. Xie and E. P. Xing, 'On the automatic generation of medical imaging reports,' *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 2577–2586, 2018. DOI: [10.18653/v1/p18-1240](https://doi.org/10.18653/v1/p18-1240).
- [33] Y. Xue, T. Xu, L. R. Long *et al.*, 'Multimodal recurrent model with attention for automated radiology report generation,' in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2018, pp. 457–466.
- [34] X. Wang, Y. Peng, L. Lu, Z. Lu and R. M. Summers, 'Tienet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9049–9058.
- [35] R. M. Thanki and A. Kothari, 'Data compression and its application in medical imaging,' in *Hybrid and Advanced Compression Techniques for Medical Images*, Springer, 2019, pp. 1–15.
- [36] H. Y. F. Wong, H. Y. S. Lam, A. H.-T. Fong *et al.*, 'Frequency and distribution of chest radiographic findings in patients positive for COVID-19,' *Radiology*, vol. 296, no. 2, E72–E78, 2020, ISSN: 0033-8419.
- [37] A. C. R. Radiology, 'ACR recommendations for the use of chest radiography and computed tomography (CT) for suspected COVID-19. Infection,' *ACR website.*, 2020.
- [38] B. K. Sahu and R. Verma, 'DICOM search in medical image archive solution e-Sushrut Chhavi,' in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, IEEE, 2011, pp. 256–260, ISBN: 1424486793.
- [39] O. Six and B. V. Quantib, *The ultimate guide to AI in radiology*, 2018.

- [40] S. M. Lee, J. B. Seo, J. Yun *et al.*, ‘Deep learning applications in chest radiography and computed tomography,’ *Journal of thoracic imaging*, vol. 34, no. 2, pp. 75–85, 2019, ISSN: 0883-5993.
- [41] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty and A.-B. M. Salem, ‘Classification using deep learning neural networks for brain tumors,’ *Future Computing and Informatics Journal*, vol. 3, no. 1, pp. 68–71, 2018, ISSN: 2314-7288.
- [42] C. Lam, C. Yu, L. Huang and D. Rubin, ‘Retinal lesion detection with deep learning using image patches,’ *Investigative ophthalmology & visual science*, vol. 59, no. 1, pp. 590–596, 2018, ISSN: 1552-5783.
- [43] G. Wang, W. Li, M. A. Zuluaga *et al.*, ‘Interactive medical image segmentation using deep learning with image-specific fine tuning,’ *IEEE transactions on medical imaging*, vol. 37, no. 7, pp. 1562–1573, 2018, ISSN: 0278-0062.
- [44] A. Qayyum, S. M. Anwar, M. Awais and M. Majid, ‘Medical image retrieval using deep convolutional neural network,’ *Neurocomputing*, vol. 266, pp. 8–20, 2017, ISSN: 0925-2312.
- [45] A. S. Chaudhari, Z. Fang, F. Kogan *et al.*, ‘Super-resolution musculoskeletal MRI using deep learning,’ *Magnetic resonance in medicine*, vol. 80, no. 5, pp. 2139–2154, 2018, ISSN: 0740-3194.
- [46] E. Gibson, W. Li, C. Sudre *et al.*, ‘NiftyNet: a deep-learning platform for medical imaging,’ *Computer methods and programs in biomedicine*, vol. 158, pp. 113–122, 2018, ISSN: 0169-2607.
- [47] M. Biswas, V. Kuppili, L. Saba *et al.*, ‘State-of-the-art review on deep learning in medical imaging.,’ *Frontiers in bioscience (Landmark edition)*, vol. 24, pp. 392–426, 2019, ISSN: 1093-4715.
- [48] S. N. Jain, T. Modi, Y. Aswani and R. U. Varma, ‘Chest radiography in adult critical care unit: A pictorial review,’ *Indian Journal of Radiology and Imaging*, vol. 29, no. 04, pp. 418–425, 2019, ISSN: 0971-3026.
- [49] E. Eisenhuber, C. M. Schaefer-Prokop, H. Prosch and W. Schima, ‘Bedside chest radiography,’ *Respiratory Care*, vol. 57, no. 3, pp. 427–443, 2012, ISSN: 0020-1324.

- [50] S. Rakshit, I. Saha, M. Wlasnowolski, U. Maulik and D. Plewczynski, 'Deep learning for detection and localization of thoracic diseases using chest x-ray imagery,' in *International Conference on Artificial Intelligence and Soft Computing*, Springer, 2019, pp. 271–282.
- [51] H. Behzadi-Khormouji, H. Rostami, S. Salehi *et al.*, 'Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images,' *Computer methods and programs in biomedicine*, vol. 185, p. 105 162, 2020, ISSN: 0169-2607.
- [52] D. M. Hansell, A. A. Bankier, H. MacMahon, T. C. McLoud, N. L. Müller and J. Remy, 'Fleischner Society: Glossary of terms for thoracic imaging,' *Radiology*, vol. 246, no. 3, pp. 697–722, Mar. 2008, ISSN: 00338419. DOI: [10.1148/RADIOLOGY.2462070712](https://doi.org/10.1148/RADIOLOGY.2462070712).
- [53] H. Behzadi-Khormouji, H. Rostami, S. Salehi *et al.*, 'Deep learning, reusable and problem-based architectures for detection of consolidation on chest x-ray images,' *Computer methods and programs in biomedicine*, vol. 185, p. 105 162, 2020.
- [54] J. Zhong, J. Tang, C. Ye and L. Dong, 'The immunology of COVID-19: is immune modulation an option for treatment?' *The Lancet Rheumatology*, vol. 2, no. 7, e428–e436, 2020, ISSN: 2665-9913.
- [55] T. Singhal, 'A review of coronavirus disease-2019 (COVID-19),' *The Indian Journal of Pediatrics*, pp. 1–6, 2020, ISSN: 0973-7693.
- [56] K. P. Trayes, J. Studdiford, S. Pickle and A. S. Tully, 'Edema: diagnosis and management,' *American family physician*, vol. 88, no. 2, pp. 102–110, 2013, ISSN: 0002-838X.
- [57] X. Wang, E. Schwab, J. Rubin *et al.*, 'Pulmonary Edema Severity Estimation in Chest Radiographs Using Deep Learning.,' in *International Conference on Medical Imaging with Deep Learning—Extended Abstract Track*, 2019, pp. 1–5.
- [58] *Heart Failure* | American Heart Association. [Online]. Available: <https://www.heart.org/en/health-topics/heart-failure> (visited on 29/08/2022).
- [59] J.-B. Lamare, T. Olatunji and L. Yao, 'On the diminishing return of labeling clinical reports,' *arXiv preprint arXiv:2010.14587*, 2020.
- [60] F. Gaillard, *Radiopaedia.org, the wiki-based collaborative Radiology resource*, 2014. [Online]. Available: <http://radiopaedia.org/> (visited on 31/08/2022).

- [61] D. H. Livingston, B. Shogan, P. John and R. F. Lavery, 'Ct diagnosis of rib fractures and the prediction of acute respiratory failure,' *Journal of Trauma and Acute Care Surgery*, vol. 64, no. 4, pp. 905–911, 2008.
- [62] S. Xu, J. Guo, G. Zhang and R. Bie, 'Automated detection of multiple lesions on chest X-ray images: Classification using a neural network technique with association-specific contexts,' *Applied Sciences*, vol. 10, no. 5, p. 1742, 2020, ISSN: 2076-3417.
- [63] M. H. Antor, 'Lung opacity identification using mathematical model based on deep learning,' *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 5, pp. 25–29, 2020.
- [64] M. Noppen, 'Spontaneous pneumothorax: Epidemiology, pathophysiology and cause,' *European Respiratory Review*, vol. 19, no. 117, pp. 217–219, 2010.
- [65] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao and R. M. Summers, 'Interleaved text/image deep mining on a very large-scale radiology database,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1090–1099.
- [66] Y. Dong, Y. Pan, J. Zhang and W. Xu, 'Learning to read chest X-ray images from 16000+ examples using CNN,' in *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, IEEE, 2017, pp. 51–57, ISBN: 1509047220.
- [67] S. S. Mousavi, M. Schukat and E. Howley, 'Deep Reinforcement Learning: An Overview,' *Lecture Notes in Networks and Systems*, vol. 16, pp. 426–440, 2018, ISSN: 23673389.
- [68] B. J. Erickson, P. Korfiatis, T. L. Kline, Z. Akkus, K. Philbrick and A. D. Weston, 'Deep learning in radiology: does one size fit all?' *Journal of the American College of Radiology*, vol. 15, no. 3, pp. 521–526, 2018, ISSN: 1546-1440.
- [69] D. A. Clevert, T. Unterthiner and S. Hochreiter, 'Fast and accurate deep network learning by exponential linear units (ELUs),' *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.
- [70] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, 'Activation functions: Comparison of trends in practice and research for deep learning,' *arXiv preprint arXiv:1811.03378*, 2018.

- [71] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, ‘Gradient-based learning applied to document recognition,’ *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, ISSN: 0018-9219.
- [72] X. Wang, L. Lu, H.-C. Shin *et al.*, ‘Unsupervised category discovery via looped deep pseudo-task optimization using a large scale radiology image database,’ *arXiv preprint arXiv:1603.07965*, 2016.
- [73] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. De Vries, M. J. N. L. Benders and I. Išgum, ‘Automatic segmentation of MR brain images with a convolutional neural network,’ *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1252–1261, 2016, ISSN: 0278-0062.
- [74] A. Krizhevsky, I. Sutskever and G. E. Hinton, ‘Imagenet classification with deep convolutional neural networks,’ *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
- [75] M. D. Zeiler and R. Fergus, ‘Visualizing and understanding convolutional networks,’ in *European conference on computer vision*, Springer, 2014, pp. 818–833.
- [76] K. Simonyan and A. Zisserman, ‘Very deep convolutional networks for large-scale image recognition,’ *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [77] C. Szegedy, W. Liu, Y. Jia *et al.*, ‘Going deeper with convolutions,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [78] K. He, X. Zhang, S. Ren and J. Sun, ‘Deep residual learning for image recognition,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [79] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, ‘Aggregated residual transformations for deep neural networks,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.
- [80] *ImageNet*. [Online]. Available: <https://image-net.org/challenges/LSVRC/2016/> (visited on 23/05/2021).
- [81] J. Hu, L. Shen and G. Sun, ‘Squeeze-and-excitation networks,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

- [82] P. Stock and M. Cisse, ‘Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases,’ in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 498–512.
- [83] X. Glorot and Y. Bengio, ‘Understanding the difficulty of training deep feedforward neural networks,’ in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [84] C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi, ‘Inception-v4, inception-resnet and the impact of residual connections on learning,’ in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.
- [85] M. Lin, Q. Chen and S. Yan, ‘Network in network,’ *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.
- [86] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, ‘Densely connected convolutional networks,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [87] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao and R. M. Summers, ‘Interleaved text/image deep mining on a large-scale radiology database for automated image interpretation,’ *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3729–3759, 2016, ISSN: 1532-4435.
- [88] J. Yarnall, *X-Ray Classification Using Deep Learning and the MIMIC-CXR Dataset*, 2020.
- [89] R. J. Williams and D. Zipser, ‘A learning algorithm for continually running fully recurrent neural networks,’ *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989, ISSN: 0899-7667.
- [90] T. Mikolov, M. Karafiát, L. Burget, J. Černocký and S. Khudanpur, ‘Recurrent neural network based language model,’ in *Eleventh annual conference of the international speech communication association*, 2010.
- [91] S. Hochreiter and J. Schmidhuber, ‘Long short-term memory,’ *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, ISSN: 0899-7667.

- [92] K. Cho, B. Van Merriënboer, C. Gulcehre *et al.*, ‘Learning phrase representations using RNN encoder-decoder for statistical machine translation,’ *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014.
- [93] C. Shorten and T. M. Khoshgoftaar, ‘A survey on image data augmentation for deep learning,’ *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019, ISSN: 2196-1115.
- [94] S. Calderon-Ramirez, S. Yang, A. Moemeni *et al.*, ‘Correcting data imbalance for semi-supervised COVID-19 detection using X-ray chest images,’ *Applied Soft Computing*, vol. 111, 2021, ISSN: 15684946.
- [95] L. Taylor and G. Nitschke, ‘Improving Deep Learning with Generic Data Augmentation,’ *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018*, pp. 1542–1547, 2019.
- [96] L. F. Rodrigues, M. C. Naldi and J. F. Mari, ‘Comparing convolutional neural networks and preprocessing techniques for HEP-2 cell classification in immunofluorescence images,’ *Computers in biology and medicine*, vol. 116, p. 103 542, 2020, ISSN: 0010-4825.
- [97] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, ‘Optuna: A next-generation hyperparameter optimization framework,’ in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.
- [98] M. Nishio, K. Fujimoto and K. Togashi, ‘Lung segmentation on chest X-ray images in patients with severe abnormal findings using deep learning,’ *International Journal of Imaging Systems and Technology*, 2020, ISSN: 0899-9457.
- [99] M. Abadi, P. Barham, J. Chen *et al.*, ‘Tensorflow: A system for large-scale machine learning,’ in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283, ISBN: 1931971331.
- [100] Y. Wu *et al.*, *Tensorpack*, <https://github.com/tensorpack/>, 2016.
- [101] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

- [102] Y. Jia, E. Shelhamer, J. Donahue *et al.*, ‘Caffe: Convolutional architecture for fast feature embedding,’ in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.
- [103] N. Ketkar, ‘Introduction to pytorch,’ in *Deep learning with python*, Springer, 2017, pp. 195–208.
- [104] A. Paszke, S. Gross, S. Chintala *et al.*, ‘Automatic differentiation in pytorch,’ 2017.
- [105] J. Howard and S. Gugger, ‘Fastai: A layered API for deep learning,’ *Information*, vol. 11, no. 2, p. 108, 2020.
- [106] J. Rubin, D. Sanghavi, C. Zhao, K. Lee, A. Qadir and M. Xu-Wilson, ‘Large scale automated reading of frontal and lateral chest x-rays using dual convolutional neural networks,’ *arXiv preprint arXiv:1804.07839*, 2018.
- [107] P. Kisilev, E. Sason, E. Barkan and S. Hashoul, ‘Medical image description using multi-task-loss CNN,’ in *Deep Learning and Data Labeling for Medical Applications*, Springer, 2016, pp. 121–129.
- [108] P. Rajpurkar, J. Irvin, K. Zhu *et al.*, ‘Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning,’ *arXiv preprint arXiv:1711.05225*, 2017.
- [109] J. Howard and S. Gugger, *Deep Learning for Coders with fastai and PyTorch*. O’Reilly Media, 2020.
- [110] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri and R. M. Summers, ‘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, pp. 2097–2106.
- [111] P. K. Gyawali, Z. Li, S. Ghimire and L. Wang, ‘Semi-supervised learning by disentangling and self-ensembling over stochastic latent space,’ in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2019, pp. 766–774.
- [112] L. Seyyed-Kalantari, G. Liu, M. McDermott, I. Y. Chen and M. Ghassemi, ‘CheX-clusion: Fairness gaps in deep chest X-ray classifiers,’ in *BIOCOMPUTING 2021: Proceedings of the Pacific Symposium*, World Scientific, 2020, pp. 232–243.

- [113] A. E. W. Johnson, T. J. Pollard, S. J. Berkowitz *et al.*, ‘MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports,’ *Scientific Data*, vol. 6, 2019.
- [114] D. Hou, Z. Zhao and S. Hu, ‘Multi-label learning with visual-semantic embedded knowledge graph for diagnosis of radiology imaging,’ *IEEE Access*, vol. 9, pp. 15 720–15 730, 2021, ISSN: 2169-3536.
- [115] A. Bustos, A. Pertusa, J.-M. Salinas and M. de la Iglesia-Vayá, ‘Padchest: A large chest x-ray image dataset with multi-label annotated reports,’ *Medical image analysis*, vol. 66, p. 101 797, 2020, ISSN: 1361-8415.
- [116] H. Bertrand, M. Hashir and J. P. Cohen, ‘Do lateral views help automated chest x-ray predictions?’ In *International Conference on Medical Imaging with Deep Learning–Extended Abstract Track*, 2019.
- [117] J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Duong and M. Ghassemi, ‘Covid-19 image data collection: Prospective predictions are the future,’ *MELBA*, p. 18 272, 2020.
- [118] L. Wang, Z. Q. Lin and A. Wong, ‘Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images,’ *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.
- [119] M. M. A. Monshi, J. Poon, V. Chung and F. M. Monshi, ‘CovidXrayNet: Optimizing Data Augmentation and CNN Hyperparameters for Improved COVID-19 Detection from CXR,’ *Computers in Biology and Medicine*, vol. 133, no. 0010-4825, p. 104 375, 2021.
- [120] A. E. W. Johnson, T. J. Pollard, N. R. Greenbaum *et al.*, ‘MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs,’ *preprint arXiv:1901.07042*, 2019.
- [121] Y. Peng, X. Wang, L. Lu, M. Bagheri, R. Summers and Z. Lu, ‘Negbio: a high-performance tool for negation and uncertainty detection in radiology reports,’ *AMIA Summits on Translational Science Proceedings*, vol. 2018, p. 188, 2018.

- [122] O. Bodenreider, ‘The unified medical language system (UMLS) integrating biomedical terminology,’ *Nucleic acids research*, vol. 32, no. suppl-1, pp. D267–D270, 2004, ISSN: 0305-1048.
- [123] J. Bridge, Y. Meng, Y. Zhao *et al.*, ‘Introducing the GEV Activation Function for Highly Unbalanced Data to Develop COVID-19 Diagnostic Models,’ *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2776–2786, 2020, ISSN: 2168-2194.
- [124] L. Wang, A. Wong, Z. Q. Lin *et al.*, ‘Figure 1 covid-19 chest x-ray dataset initiative,’ *Accessed: May*, vol. 9, 2020.
- [125] Y. Peng, Y. Tang, S. Lee, Y. Zhu, R. M. Summers and Z. Lu, ‘COVID-19-CT-CXR: A freely accessible and weakly labeled chest X-Ray and CT image collection on COVID-19 from biomedical literature,’ *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 3–12, 2021, ISSN: 23327790.
- [126] M. Heath, K. Bowyer, D. Kopans, R. Moore and W. P. Kegelmeyer, ‘The digital database for screening mammography Proc. 5th Int,’ in *Workshop on Digital Mammography*, 2000, pp. 212–218.
- [127] K. N. Jones, D. E. Woode, K. Panizzi and P. G. Anderson, ‘Peir digital library: Online resources and authoring system,’ in *Proceedings of the AMIA Symposium*, American Medical Informatics Association, 2001, p. 1075.
- [128] L. Deng and Y. Liu, ‘A joint introduction to natural language processing and to deep learning,’ in *Deep learning in natural language processing*, Springer, 2018, pp. 1–22.
- [129] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, ‘Natural language processing (almost) from scratch,’ *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.
- [130] M. B. A. McDermott, T. M. H. Hsu, W.-H. Weng, M. Ghassemi and P. Szolovits, ‘CheXpert++: Approximating the CheXpert Labeler for Speed, Differentiability, and Probabilistic Output,’ in *Machine Learning for Healthcare Conference*, PMLR, 2020, pp. 913–927.
- [131] A. Smit, S. Jain, P. Rajpurkar, A. Pareek, A. Y. Ng and M. P. Lungren, ‘CheXbert: Combining automatic labelers and expert annotations for accurate radiology report

- labeling using BERT,' *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1500–1519, 2020.
- [132] M. M. A. Monshi, J. Poon, V. Chung and F. M. Monshi, 'Labeling Chest X-Ray Reports Using Deep Learning,' in *International Conference on Artificial Neural Networks*, Springer, 2021, pp. 684–694.
- [133] A. R. Aronson and F.-M. Lang, 'An overview of MetaMap: historical perspective and recent advances,' *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 229–236, 2010, ISSN: 1527-974X.
- [134] R. Leaman, R. Islamaj Doğan and Z. Lu, 'DNorm: disease name normalization with pairwise learning to rank,' *Bioinformatics*, vol. 29, no. 22, pp. 2909–2917, 2013, ISSN: 1460-2059.
- [135] L. Oakden-Rayner, 'Exploring large-scale public medical image datasets,' *Academic Radiology*, vol. 27, no. 1, pp. 106–112, 2020, ISSN: 1076-6332.
- [136] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, 'BERT: Pre-training of deep bidirectional transformers for language understanding,' *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 2019.
- [137] E. Alsentzer, J. Murphy, W. Boag *et al.*, 'Publicly available clinical bert embeddings,' in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 2019, pp. 72–78.
- [138] J. Mullenbach, S. Wiegrefe, J. Duke, J. Sun and J. Eisenstein, 'Explainable Prediction of Medical Codes from Clinical Text,' in *NAACL-HLT*, 2018.
- [139] L. Deng and Y. Liu, *Deep learning in natural language processing*. Springer, 2018, ISBN: 9811052093.
- [140] X. He and L. Deng, 'Deep learning in natural language generation from images,' in *Deep learning in natural language processing*, Springer, 2018, pp. 289–307.
- [141] S. Hassanpour and C. P. Langlotz, 'Unsupervised topic modeling in a large free text radiology report repository,' *Journal of digital imaging*, vol. 29, no. 1, pp. 59–62, 2016, ISSN: 0897-1889.

- [142] H.-C. Shin, L. Lu and R. M. Summers, ‘Natural language processing for large-scale medical image analysis using deep learning,’ *Deep learning for medical image analysis*, pp. 405–421, 2017.
- [143] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, ‘MixUp: Beyond empirical risk minimization,’ *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.
- [144] L. Yao, E. Poblenz, D. Dagunts, B. Covington, D. Bernard and K. Lyman, ‘Learning to diagnose from scratch by exploiting dependencies among labels,’ *arXiv preprint arXiv:1710.10501*, 2017.
- [145] H. Wang, H. Jia, L. Lu and Y. Xia, ‘Thorax-Net: An Attention Regularized Deep Neural Network for Classification of Thoracic Diseases on Chest Radiography,’ *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 475–485, 2019, ISSN: 21682208.
- [146] M. M. A. Monshi, J. Poon and V. Chung, ‘Convolutional Neural Network to Detect Thorax Diseases from Multi-view Chest X-Rays,’ in *International Conference on Neural Information Processing*, Springer, 2019, pp. 148–158.
- [147] Q. Guan, Y. Huang, Z. Zhong, Z. Zheng, L. Zheng and Y. Yang, ‘Thorax disease classification with attention guided convolutional neural network,’ *Pattern Recognition Letters*, vol. 131, pp. 38–45, 2020, ISSN: 01678655.
- [148] E. E.-D. Hemdan, M. A. Shouman and M. E. Karar, ‘Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images,’ *arXiv preprint arXiv:2003.11055*, 2020.
- [149] T. Mahmud, M. A. Rahman and S. A. Fattah, ‘CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization,’ *Computers in biology and medicine*, vol. 122, p. 103 869, 2020, ISSN: 0010-4825.
- [150] A. Narin, C. Kaya and Z. Pamuk, ‘Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks,’ *Pattern Analysis and Applications*, vol. 24, no. 3, pp. 1207–1220, 2021, ISSN: 1433755X.

- [151] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim and U. R. Acharya, 'Automated detection of COVID-19 cases using deep neural networks with X-ray images,' *Computers in Biology and Medicine*, p. 103 792, 2020, ISSN: 0010-4825.
- [152] M. M. A. Monshi and J. Poon, 'Distributed Deep Learning for Multi-Label Chest Radiography Classification,' *Visigrapp*, vol. 4, 2022, pp. 949–956, ISBN: 978-989-758-555-5.
- [153] M. Nishio, S. Noguchi, H. Matsuo and T. Murakami, 'Automatic classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray image: combination of data augmentation methods,' *Scientific reports*, vol. 10, no. 1, pp. 1–6, 2020, ISSN: 2045-2322.
- [154] I. D. Apostolopoulos and T. A. Mpesiana, 'Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks,' *Physical and Engineering Sciences in Medicine*, p. 1, 2020.
- [155] P. K. Sethy, S. K. Behera, P. K. Ratha and P. Biswas, 'Detection of coronavirus disease (covid-19) based on deep features and support vector machine,' *International Journal of Mathematical Engineering and Management Sciences*, pp. 643–651, 2020.
- [156] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization,' in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
- [157] M. Hashir, H. Bertrand and J. P. Cohen, 'Quantifying the value of lateral views in deep learning for chest x-rays,' in *Medical Imaging with Deep Learning*, PMLR, 2020, pp. 288–303.
- [158] P. Mooney, 'Chest x-ray images (pneumonia),' 2018.
- [159] Z. Wang, Y. Xiao, Y. Li *et al.*, 'Automatically discriminating and localizing COVID-19 from community-acquired pneumonia on chest X-rays,' *Pattern Recognition*, p. 107 613, 2020, ISSN: 0031-3203.
- [160] F. Wang, M. Jiang, C. Qian *et al.*, 'Residual attention network for image classification,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3156–3164.

- [161] J. Redmon and A. Farhadi, 'YOLO9000: better, faster, stronger,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [162] A. Wong, M. J. Shafiee, B. Chwyl and F. Li, 'Ferminets: Learning generative machines to generate efficient neural networks via generative synthesis,' *arXiv preprint arXiv:1809.05989*, 2018.
- [163] Y. Oh, S. Park and J. C. Ye, 'Deep Learning COVID-19 Features on CXR using Limited Training Data Sets,' *IEEE Transactions on Medical Imaging*, pp. 1–1, 2020, ISSN: 0278-0062.
- [164] A. G. Howard *et al.*, 'Mobilenets: Efficient convolutional neural networks for mobile vision applications,' *arXiv preprint arXiv:1704.04861*, 2017.
- [165] D. M. Blei, A. Y. Ng and M. I. Jordan, 'Latent dirichlet allocation,' *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003, ISSN: 1532-4435.
- [166] A. Karpathy and L. Fei-Fei, 'Deep visual-semantic alignments for generating image descriptions,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.
- [167] K. Xu, J. Ba, R. Kiros *et al.*, 'Show, attend and tell: Neural image caption generation with visual attention,' in *International conference on machine learning*, PMLR, 2015, pp. 2048–2057.
- [168] J. Guo, X. Yuan, X. Zheng, P. Xu, Y. Xiao and B. Liu, 'Diagnosis labeling with disease-specific characteristics mining,' *Artificial intelligence in medicine*, vol. 100, no. 90, pp. 25–33, 2018, ISSN: 0933-3657.
- [169] C. Qin, D. Yao, Y. Shi and Z. Song, 'Computer-aided detection in chest radiography based on artificial intelligence: a survey,' *Biomedical engineering online*, vol. 17, no. 1, pp. 1–23, 2018, ISSN: 1475-925X.
- [170] S. Candemir, S. Jaeger, K. Palaniappan *et al.*, 'Lung segmentation in chest radiographs using anatomical atlases with nonrigid registration,' *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 577–590, 2013.
- [171] S. Candemir, S. Jaeger, W. Lin, Z. Xue, S. Antani and G. Thoma, 'Automatic heart localization and radiographic index computation in chest x-rays,' in *Medical Imaging 2016: Computer-Aided Diagnosis*, SPIE, vol. 9785, 2016, pp. 302–309.

- [172] H. X. Nguyen and T. T. Dang, 'Ribs suppression in chest x-ray images by using ica method,' in *5th International Conference on Biomedical Engineering in Vietnam*, Springer, 2015, pp. 194–197.
- [173] E. Çallı, E. Sogancioglu, B. van Ginneken, K. G. van Leeuwen and K. Murphy, 'Deep Learning for Chest X-ray Analysis: A Survey,' *Medical Image Analysis*, p. 102 125, 2021, ISSN: 1361-8415.
- [174] D. Yu, K. Zhang, L. Huang *et al.*, 'Detection of peripherally inserted central catheter (picc) in chest x-ray images: A multi-task deep learning model,' *Computer Methods and Programs in Biomedicine*, vol. 197, p. 105 674, 2020.
- [175] J. Wessel, M. P. Heinrich, J. von Berg, A. Franz and A. Saalbach, 'Sequential rib labeling and segmentation in chest x-ray using mask r-cnn,' in *International Conference on Medical Imaging with Deep Learning—Extended Abstract Track*, 2019.
- [176] Y. Cho, Y.-G. Kim, S. M. Lee, J. B. Seo and N. Kim, 'Reproducibility of abnormality detection on chest radiographs using convolutional neural network in paired radiographs obtained within a short-term interval,' *Scientific Reports*, vol. 10, no. 1, pp. 1–11, 2020.
- [177] Y. Karbhari, A. Basu, Z. W. Geem, G.-T. Han and R. Sarkar, 'Generation of synthetic chest x-ray images and detection of covid-19: A deep learning based approach,' *Diagnostics*, vol. 11, no. 5, p. 895, 2021.
- [178] J. Goodfellow Ian, P.-A. Jean, M. Mehdi *et al.*, 'Generative adversarial nets,' in *Proceedings of the 27th international conference on neural information processing systems*, vol. 2, 2014, pp. 2672–2680.
- [179] X. Yi, E. Walia and P. Babyn, 'Generative adversarial network in medical imaging: A review,' *Medical Image Analysis*, vol. 58, p. 101 552, 2019, ISSN: 1361-8415.
- [180] T. Fawcett, 'An introduction to ROC analysis,' *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006, ISSN: 0167-8655.
- [181] J. Gorodkin, 'Comparing two K-category assignments by a K-category correlation coefficient,' *Computational biology and chemistry*, vol. 28, no. 5-6, pp. 367–374, 2004, ISSN: 1476-9271.

- [182] M. Grandini, E. Bagli and G. Visani, ‘Metrics for Multi-Class Classification: an Overview,’ *arXiv preprint arXiv:2008.05756*, 2020.
- [183] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, ‘BLEU: a method for automatic evaluation of machine translation,’ in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [184] C.-Y. Lin, ‘Rouge: A package for automatic evaluation of summaries,’ in *Text summarization branches out*, 2004, pp. 74–81.
- [185] M. Denkowski and A. Lavie, ‘Meteor universal: Language specific translation evaluation for any target language,’ in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 376–380.
- [186] R. Vedantam, C. Lawrence Zitnick and D. Parikh, ‘Cider: Consensus-based image description evaluation,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.
- [187] P. Anderson, B. Fernando, M. Johnson and S. Gould, ‘Spice: Semantic propositional image caption evaluation,’ in *European conference on computer vision*, Springer, 2016, pp. 382–398.
- [188] C.-Y. Lin and F. J. Och, ‘Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics,’ in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 2004, pp. 605–612.
- [189] M. Kilickaya, A. Erdem, N. Ikizler-Cinbis and E. Erdem, ‘Re-evaluating automatic metrics for image captioning,’ *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 1, pp. 199–209, 2017.
- [190] S. A. Hicks, K. Pogorelov, T. de Lange *et al.*, ‘Comprehensible reasoning and automated reporting of medical examinations based on deep learning analysis,’ in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 490–493.
- [191] J. Mańdziuk and A. Żychowski, ‘Dimensionality Reduction in Multilabel Classification with Neural Networks,’ in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–8, ISBN: 1728119855.

- [192] V. Liventsev, I. Fedulova and D. Dyllov, 'Deep text prior: Weakly supervised learning for assertion classification,' in *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 243–257.
- [193] S. Merity, N. S. Keskar and R. Socher, 'Regularizing and optimizing LSTM language models,' *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.
- [194] S. Merity, C. Xiong, J. Bradbury and R. Socher, 'Pointer sentinel mixture models,' *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.
- [195] J. Howard and S. Ruder, 'Universal language model fine-tuning for text classification,' in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.
- [196] S. Ruder, *Neural transfer learning for natural language processing*, 2019.
- [197] S. Harsha Kadam and K. Paniskaki, 'Text analysis for email multi label classification,' *Open Digital Repository*, 2020.
- [198] D. Ganeshan, P.-A. T. Duong, L. Probyn *et al.*, 'Structured reporting in radiology,' *Academic radiology*, vol. 25, no. 1, pp. 66–73, 2018, ISSN: 1076-6332.
- [199] I. Drozdov, D. Forbes, B. Szubert, M. Hall, C. Carlin and D. J. Lowe, 'Supervised and unsupervised language modelling in Chest X-Ray radiological reports,' *Plos one*, vol. 15, no. 3, e0229963, 2020, ISSN: 1932-6203.
- [200] X. Huang, Y. Fang, M. Lu, Y. Yao and M. Li, 'An annotation model on end-to-end chest radiology reports,' *IEEE Access*, vol. 7, pp. 65 757–65 765, 2019, ISSN: 2169-3536.
- [201] S. Datta, Y. Si, L. Rodriguez, S. E. Shooshan, D. Demner-Fushman and K. Roberts, 'Understanding spatial language in radiology: Representation framework, annotation, and spatial relation extraction from chest X-ray reports using deep learning,' *Journal of biomedical informatics*, vol. 108, p. 103 473, 2020, ISSN: 1532-0464.
- [202] P. Kordjamshidi, M. Van Otterlo and M.-F. Moens, 'Spatial role labeling: Towards extraction of spatial relations from natural language,' *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 8, no. 3, pp. 1–36, 2011, ISSN: 1550-4875.

- [203] S. Jain, A. Smit, A. Y. Ng and P. Rajpurkar, ‘Effect of Radiology Report Labeler Quality on Deep Learning Models for Chest X-Ray Interpretation,’ *arXiv preprint arXiv:2104.00793*, 2021.
- [204] S. Jain, A. Smit, S. Q. Truong *et al.*, ‘VisualCheXbert: Addressing the discrepancy between radiology report labels and image labels,’ *ACM CHIL 2021 - Proceedings of the 2021 ACM Conference on Health, Inference, and Learning*, pp. 105–115, 2021.
- [205] L. N. Smith, ‘A disciplined approach to neural network hyper-parameters: Part 1—learning rate, batch size, momentum, and weight decay,’ *arXiv preprint arXiv:1803.09820*, 2018.
- [206] Z. Hussain, F. Gimenez, D. Yi and D. Rubin, ‘Differential data augmentation techniques for medical imaging classification tasks,’ in *AMIA Annual Symposium Proceedings*, vol. 2017, American Medical Informatics Association, 2017, p. 979.
- [207] J. A. Dunnmon, D. Yi, C. P. Langlotz, C. Ré, D. L. Rubin and M. P. Lungren, ‘Assessment of convolutional neural networks for automated classification of chest radiographs,’ *Radiology*, vol. 290, no. 2, pp. 537–544, 2019, ISSN: 0033-8419.
- [208] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu *et al.*, ‘Convolutional neural networks for medical image analysis: Full training or fine tuning?’ *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016, ISSN: 0278-0062.
- [209] L. N. Smith and N. Topin, ‘Super-convergence: Very fast training of neural networks using large learning rates,’ in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006, SPIE, 2019, pp. 369–386.
- [210] C. Coleman, D. Kang, D. Narayanan *et al.*, ‘Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark,’ *ACM SIGOPS Operating Systems Review*, vol. 53, no. 1, pp. 14–25, 2019, ISSN: 0163-5980.
- [211] E. T. Nader, ‘Chest X-ray interpretation,’ in *Perioperative Assessment of the Maxillo-facial Surgery Patient*, Springer, 2018, pp. 119–127.
- [212] W. Pezzotti, ‘Chest X-ray interpretation: not just black and white,’ *Nursing2020*, vol. 44, no. 1, pp. 40–47, 2014, ISSN: 0360-4039.
- [213] V. Ku, ‘A fresh look at chest x-rays,’ *Nursing2020 Critical Care*, vol. 7, no. 6, pp. 23–29, 2012.

- [214] H. H. Pham, T. T. Le, D. Q. Tran, D. T. Ngo and H. Q. Nguyen, ‘Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels,’ *Neurocomputing*, vol. 437, pp. 186–194, 2021, ISSN: 0925-2312.
- [215] B. Chen, J. Li, G. Lu, H. Yu and D. Zhang, ‘Label co-occurrence learning with graph convolutional networks for multi-label chest x-ray image classification,’ *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, pp. 2292–2302, 2020, ISSN: 2168-2194.
- [216] ‘NVIDIA DGX A100 System Architecture,’ 2020. [Online]. Available: <https://bit.ly/3izVeFF>.
- [217] *DGX-2 : AI Servers for Solving Complex AI Challenges* | NVIDIA, 2018. [Online]. Available: <https://www.nvidia.com/en-us/data-center/dgx-2/>.
- [218] S. Li, Y. Zhao, R. Varma *et al.*, ‘PyTorch Distributed: Experiences on Accelerating Data Parallel Training,’ *Proceedings of the VLDB Endowment*, vol. 13, no. 12, 2020.
- [219] C. F. Sabottke and B. M. Spieler, ‘The effect of image resolution on deep learning in radiography,’ *Radiology: Artificial Intelligence*, vol. 2, no. 1, e190015, 2020, ISSN: 2638-6100.
- [220] T. Dratsch, M. Korenkov, D. Zopfs *et al.*, ‘Practical applications of deep learning: classifying the most common categories of plain radiographs in a PACS using a neural network,’ *European Radiology*, vol. 31, no. 4, pp. 1812–1818, 2021, ISSN: 1432-1084.
- [221] S. Mo and M. Cai, ‘Deep learning based multi-label chest x-ray classification with entropy weighting loss,’ in *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, IEEE, 2019, pp. 124–127, ISBN: 1728146534.
- [222] K. K. Bressemer, L. C. Adams, C. Erxleben, B. Hamm, S. M. Niehues and J. L. Vahldiek, ‘Comparing different deep learning architectures for classification of chest radiographs,’ *Scientific reports*, vol. 10, no. 1, pp. 1–16, 2020, ISSN: 2045-2322.
- [223] R. Zhang, ‘Making convolutional networks shift-invariant again,’ in *International conference on machine learning*, PMLR, 2019, pp. 7324–7334.
- [224] R. Wightman, *Pytorch image models*, <https://github.com/rwightman/pytorch-image-models>, 2021. DOI: [10.5281/zenodo.4414861](https://doi.org/10.5281/zenodo.4414861).

- [225] M. Tan and Q. Le, 'Efficientnet: Rethinking model scaling for convolutional neural networks,' in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.
- [226] B. Chen, Y. Lu and G. Lu, 'Multi-label chest X-ray image classification via label co-occurrence learning,' in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Springer, 2019, pp. 682–693.
- [227] W. Wang *et al.*, 'Detection of SARS-CoV-2 in different types of clinical specimens,' *Jama*, vol. 323, no. 18, pp. 1843–1844, 2020, ISSN: 0098-7484.
- [228] C. P. West, V. M. Montori and P. Sampathkumar, 'COVID-19 testing: the threat of false-negative results,' in *Mayo Clinic Proceedings*, vol. 95, Elsevier, 2020, pp. 1127–1129.
- [229] M.-Y. Ng *et al.*, 'Imaging profile of the COVID-19 infection: radiologic findings and literature review,' *Radiology: Cardiothoracic Imaging*, vol. 2, no. 1, e200034, 2020, ISSN: 2638-6135.
- [230] G. D. Rubin *et al.*, 'The role of chest imaging in patient management during the COVID-19 pandemic: a multinational consensus statement from the Fleischner Society,' *Chest*, 2020, ISSN: 0012-3692.
- [231] Y. LeCun, Y. Bengio and G. Hinton, 'Deep learning,' *nature*, vol. 521, no. 7553, pp. 436–444, 2015, ISSN: 1476-4687.
- [232] M. Ahsan, M. Based, J. Haider and M. Kowalski, 'COVID-19 Detection from Chest X-ray Images Using Feature Fusion and Deep Learning,' *Sensors*, vol. 21, no. 4, p. 1480, 2021.
- [233] J. Bergstra and Y. Bengio, 'Random search for hyper-parameter optimization.,' *Journal of machine learning research*, vol. 13, no. 2, 2012, ISSN: 1532-4435.
- [234] D. P. Kingma and J. L. Ba, 'Adam: A method for stochastic optimization,' *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [235] J. De Moura, L. R. García, P. F. L. Vidal *et al.*, 'Deep convolutional approaches for the analysis of covid-19 using chest x-ray images from portable devices,' *IEEE Access*, vol. 8, pp. 195 594–195 607, 2020, ISSN: 2169-3536.

- [236] Z. Zhong, L. Zheng, G. Kang, S. Li and Y. Yang, ‘Random Erasing Data Augmentation,’ in *AAAI*, 2020, pp. 13 001–13 008.
- [237] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, ‘Mobilenetv2: Inverted residuals and linear bottlenecks,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [238] M. A. Mercioni and S. Holban, ‘Soft-clipping swish: A novel activation function for deep learning,’ in *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, IEEE, 2021, pp. 225–230.
- [239] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, ‘Rethinking the inception architecture for computer vision,’ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [240] E. D. Cubuk, B. Zoph, J. Shlens and Q. V. Le, ‘Randaugment: Practical automated data augmentation with a reduced search space,’ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.
- [241] S. Lim, I. Kim, T. Kim, C. Kim and S. Kim, ‘Fast AutoAugment,’ *Advances in Neural Information Processing Systems*, vol. 32, 2019, ISSN: 10495258.
- [242] W. Wang, D. Tran and M. Feiszli, ‘What Makes Training Multi-Modal Classification Networks Hard?’ In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 695–12 705.
- [243] A. Mitra, A. Chakravarty, N. Ghosh, T. Sarkar, R. Sethuraman and D. Sheet, ‘A systematic search over deep convolutional neural network architectures for screening chest radiographs,’ in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, IEEE, 2020, pp. 1225–1228.
- [244] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, ‘Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,’ *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, 2020, ISSN: 1573-1405.

- [245] K. Simonyan, A. Vedaldi and A. Zisserman, ‘Deep inside convolutional networks: Visualising image classification models and saliency maps,’ in *In Workshop at International Conference on Learning Representations*, Citeseer, 2014.
- [246] S. Schalekamp, W. M. Klein and K. G. van Leeuwen, ‘Current and emerging artificial intelligence applications in chest imaging: a pediatric perspective,’ *Pediatric Radiology*, pp. 1–11, 2021, ISSN: 1432-1998.
- [247] K. G. van Leeuwen, S. Schalekamp, M. J. C. M. Rutten, B. van Ginneken and M. de Rooij, ‘Artificial intelligence in radiology: 100 commercially available products and their scientific evidence,’ *European radiology*, vol. 31, no. 6, pp. 3797–3804, 2021, ISSN: 1432-1084.

APPENDIX A

Poster

MultiViewModel

Introduction

Chest X-Ray: It is the most common radiologist exams in the world that demands correct and immediate diagnosis of a patient's thorax to avoid life threatening diseases.

Problem: Certified radiologists are hard to find. Stress, fatigue and experience contribute to the quality of an examination.

Solution: Automated & precise system that can flag potentially life-threatening diseases to handle emergency cases efficiently.

Example: Convolution neural network (CNN) is a supervised deep learning model that is able to learn useful features which are beyond the limit of radiology detection.

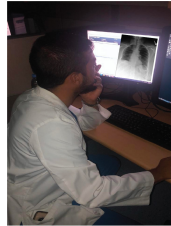


Fig 1. Radiologist Examines a Chest X-Ray

Related Work

Dataset: IU X-Ray (2015), ChestX-ray14 (2017), CheXpert, Pad-Chest & MIMIC-CXR (2019).

Model: CheXNet, text-image embedding network (TieNet) & attention guided convolutional neural network (AG-CNN).

Approach: CNN such as AlexNet, VGG-16, DenseNet & ResNet.

Gap: Using only frontal view, long training time & low accuracy.

Consistent with recent models: We focus on training CNN models to detect 12 common thorax diseases.

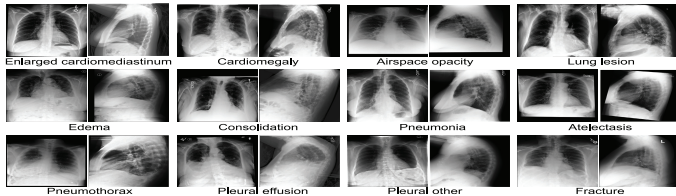


Fig 2. Examples of 12 Thoracic Diseases from MIMIC-CXR Dataset

Unique from past works: We propose a novel **stage-wise training** approach to observe the model's performance => **re-duce** training time & **increase** accuracy. We adopt a combination of **recent techniques on multi-view chest X-rays** including Res-Net-50, transfer learning, fine tuning, **fit one cycle function** & **discriminative learning rates**.

Proposed Model

Structure Overview:

We divided the task of detecting thorax diseases into 12 sub-tasks. Each task considers the presence/absence of a disease. For each binary label problem, ResNet-50 is used as the baseline CNN architecture. ResNet-50 consists of 49 convolution layers & ends with 1 fully connected layer.

Fig 3. A Basic Residual Block

last output convolution operations generated output

$$x_l = F(x_{l-1}) + x_{l-1}$$

Training Stages:

Embrace transfer learning (PyTorch & fastai)

Observe the model's performance (fit-one-cycle method)

Use the optimal learning rate finder

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

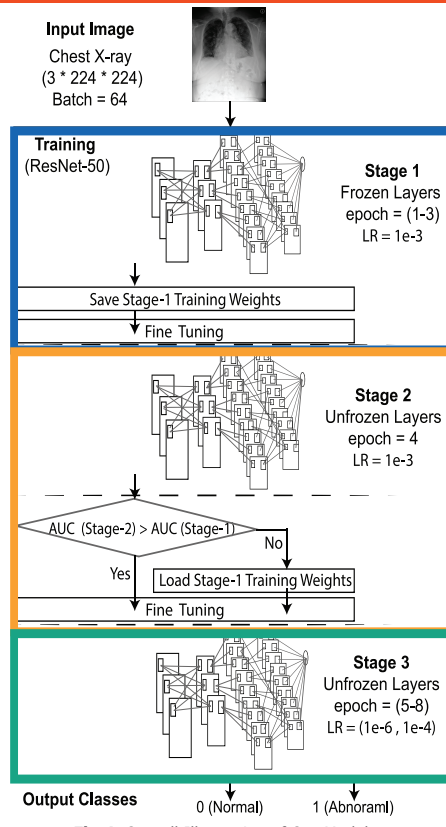


Fig 4. Overall Illustration of Our Model

Experiment

Dataset: We organized a subset of 10% of the MIMIC-CXR dataset into training set (33,195) and validation set (3,688). We dropped uncertain & unknown labels.

For example, our subset includes 6932 images with Cardiomegaly.

Pathology	Positive (%)	Negative (%)
Enlarged Cardiom.	1019 (2.8)	35367 (97.19)
Cardiomegaly	6932 (18.79)	29951 (81.2)
Airspace Opacity	7582 (20.42)	29542 (79.57)
Lung Lesion	1060 (2.82)	36472 (97.17)
Edema	3964 (11.06)	31859 (88.93)
Consolidation	1410 (3.8)	35634 (96.19)
Pneumonia	2738 (7.83)	32202 (92.16)
Atelectasis	6356 (17.54)	29876 (82.45)
Pneumothorax	1523 (4.05)	36059 (95.94)
Pleural Effusion	7869 (21.34)	28994 (78.65)
Pleural Other	425 (1.13)	37132 (98.86)
Fracture	805 (2.13)	36829 (97.86)

Pre-Processing: We employed several augmentation strategies.

Training using 224 px reduces training time without worsening AUC.

Parameter	Value
Size	224
Flip (horizontally)	True
Lighting	0.3
Affine	0.5

Image Size (pixels)	Epoch								Avg. AUC per Epoch
	1	2	3	4	5	6	7	8	
299	0.565	0.733	0.758	0.791	0.798	0.804	0.804	0.807	0.757
224	0.725	0.733	0.747	0.785	0.793	0.799	0.802	0.802	0.773

Training: We used 4 NVIDIA Tesla P4 GPUs to reduce training time.

No. of GPUs	Epoch								Avg. Time per Epoch (min)
	1	2	3	4	5	6	7	8	
1	32:42	32:26	32:36	34:34	33:40	33:52	33:58	34:00	33:28
4	13:32	12:54	13:01	13:05	13:07	13:08	13:07	13:06	13:07

Result: We computed the Area Under Curve (AUC) of each pathology on the validation set for each of the eight training epochs.

Stage 3 results in larger AUC values than **stage-1** & **stage 2** due to the discriminative learning rates.

Pathology	Epoch							
	1	2	3	4	5	6	7	8
Enlarged Cardiom.	0.670	0.694	0.700	0.544	0.702	0.705	0.708	0.710
Cardiomegaly	0.725	0.733	0.747	0.785	0.793	0.799	0.802	0.802
Airspace Opacity	0.621	0.687	0.694	0.712	0.730	0.730	0.733	0.737
Lung Lesion	0.520	0.638	0.612	0.638	0.651	0.688	0.730	0.729
Edema	0.816	0.848	0.857	0.887	0.892	0.894	0.896	0.897
Consolidation	0.748	0.758	0.769	0.778	0.788	0.797	0.797	0.799
Pneumonia	0.556	0.531	0.545	0.497	0.550	0.585	0.580	0.587
Atelectasis	0.706	0.706	0.743	0.827	0.830	0.835	0.837	0.838
Pneumothorax	0.710	0.786	0.817	0.839	0.853	0.862	0.868	0.860
Pleural Effusion	0.837	0.869	0.881	0.891	0.903	0.906	0.905	0.899
Pleural Other	0.585	0.637	0.676	0.533	0.707	0.736	0.739	0.727
Fracture	0.546	0.563	0.576	0.606	0.636	0.648	0.711	0.741
Average	0.670	0.704	0.718	0.711	0.753	0.765	0.776	0.777

In 5 out of 7 overlap pathologies, our model performs better than both DualNet models.

Pathology	DualNet		Our Model
	PA + Lateral	AP + Lateral	
Enlarged Cardiom.	-	-	0.710
Cardiomegaly	0.840	0.755	0.802
Airspace Opacity	-	-	0.737
Lung Lesion	-	-	0.730
Edema	0.734	0.749	0.897
Consolidation	0.632	0.623	0.799
Pneumonia	0.625	0.593	0.587
Atelectasis	0.766	0.671	0.838
Pneumothorax	0.706	0.621	0.868
Pleural Effusion	0.757	0.733	0.906
Pleural Other	-	-	0.739
Fracture	-	-	0.741
Average	0.722	0.677	0.779

Analysis: Our model is trained on a better annotated chest X-rays (CheXpert labeler) than DualNet (NegBio labeler). We reach improved results over those achieved by DualNet using small image sizes 224 by 224 pixels instead of 512 by 512.

Limitation: Class labels in the training set are noisy. The positive-negative subsets ratio was highly imbalanced in some pathologies. Yet, our model's AUC is above 0.7.

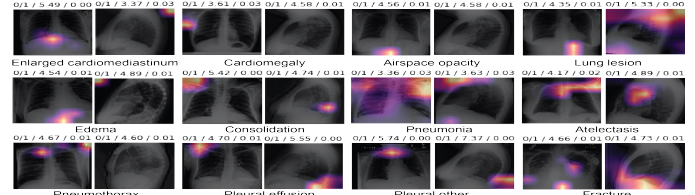


Fig 5. Examples of the Most Confused Chest X-Rays with Heatmaps

Conclusion

Contribution: We proposed ResNet-50 CNN based stage-wise models to detect 12 thorax diseases on 10% of the largest chest X-rays dataset to date, MIMIC-CXR. The absolute labelling performance with an average weighted AUC of 0.779 is encouraging.

Future Work: We plan to improve our CNN model performance through data augmentation. We will incorporate useful information from the free-text radiology reports like patient's history to accurately recognize the presence/absence of thorax diseases.

Xclassifier

Distributed Deep Learning for Multi-Label Chest Radiography Classification

Maram Monshi, Josiah Poon & Vera Chung

Problem

- Chest radiography (CXR) supports the diagnosis and treatment for a series of thoracic diseases like pneumonia
- Recent automatic classifiers use deep learning but:
 - Neglect label co-occurrence & interdependency
 - Fail to make full use of accelerators
 - Result in inefficient & computationally expensive models
- CXR classifiers' performance can be improved by:
 - Leveraging label co-occurrence (Chen et al., 2020)
 - Selecting the optimal CXR format (Sabottke and Spieler, 2020)
 - Training with an efficient approach

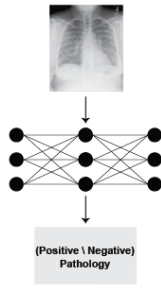


Fig 1. CXR Classification

Literature

- Binary classification with Convolution Neural Network (CNN)
 - CheXNet (Rajpurkar et al., 2017)

They did not consider pathology correlation & ignored labels relation
 - TieNet (Wang et al., 2018)
 - MultiViewModel (Monshi et al., 2019)
 - VGG16-based model (Yarnall, 2020)
- Multi-label classification
 - CheXclusion (Seyyed-Kalantari et al., 2020)
 - Latent-space self-ensemble (Hou et al., 2021)
 - VSEGCN (Hou et al., 2021)

We extended this wave of research using more efficient training methods

Method

- Dataset:
 - MIMIC-CXR (Johnson et al., 2019) 377,110 preprocessing → 356,225 CXRs
 - CheXpert (Irvin et al., 2019) 224,316 preprocessing → 212,498 CXRs
- U-zeros method
- Split (80-10-10)

Table 1: Positive label co-occurrence

Label	% of all	% of label co-occurrence													
		data	At	Ca	Co	Ed	EC	Fr	LL	LO	NF	PE	PO	Pa	Px
Atelectasis (At)	16	100	12	6	27	5	4	3	43	0	49	1	2	9	60
Cardiomegaly (Ca)	13	14	100	5	43	7	3	2	48	0	44	1	2	3	58
Consolidation (Co)	7	14	10	100	21	4	3	5	38	0	50	2	7	5	52
Edema (Ed)	25	17	22	6	100	4	2	2	53	0	51	1	2	3	64
Enlarged Cardiom. (EC)	14	18	6	20	20	100	6	5	48	0	36	2	1	7	52
Fracture (Fr)	4	14	9	4	11	7	100	4	40	0	27	3	2	12	40
Lung Lesion (LL)	4	11	7	8	9	6	4	100	58	0	36	3	5	9	35
Lung Opacity (LO)	50	13	12	5	26	3	5	100	49	2	4	9	58		
No Finding (NF)	11	0	0	0	0	0	0	0	100	0	0	0	0	39	
Pleural Effusion (PE)	41	19	14	9	31	5	3	4	61	0	100	1	2	8	61
Pleural Other (PO)	2	11	9	9	5	8	9	53	0	26	100	4	7	39	
Pneumonia (Pa)	3	10	8	17	20	3	2	8	67	0	29	2	100	2	29
Pneumothorax (Px)	9	16	4	4	8	4	5	4	47	0	34	1	100	60	
Support Devices (SD)	55	17	13	7	29	5	3	3	53	8	46	1	2	10	100

Data Augmentation:

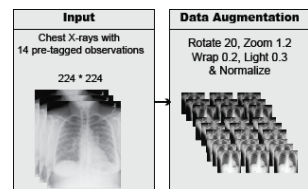


Fig 2. Data Augmentation

To increase the accuracy of detecting abnormalities from CXRs (Monshi et al., 2021)

- CNN Architecture:
 - Xclassifier is based on Dense CNN (DenseNet-121) (Huang et al., 2017)

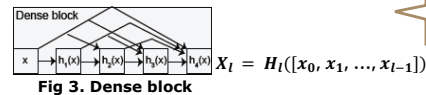


Fig 3. Dense block

Due to its success in CXRs classification (Rajpurkar et al., 2017) (Yao et al., 2017) (Mo and Cai, 2019) (Chen et al., 2020) (Bressen et al., 2020)

- Antialiasing & Subsampling:
 - Insert a blur kernel $m \times m$ before each downsampling step in DenseNet

This increases ImageNet (Zhang, 2019) & CXR classification accuracy

Table 2: DenseNet-121 variations models and training performance

Model	Description	Accuracy	AUC
DenseNet-121	Single 7x7 convolution layer with no anti-aliasing layer	90.69	81.34
DenseNet-121d	Three 3x3 convolution layers with no anti-aliasing layer	90.73	81.28
DenseNetblur-121d	Three 3x3 convolution layers with anti-aliasing blur pool	90.80	81.96

- Distributed Data Parallel (DDP): (Li et al., 2019)
 - Use 64 CXRs (batch size) for each of the 4 GPUs to accelerate convergence

DDP provides 4x speed-up over one GPU & 1.14x to 3.35x speed-up over data parallel

Fig 4. Visualizing DDP

Table 3: Training approaches and training performance

Training Approach	Dataset	Accuracy	AUC	Avg. time per epoch (min)
Single GPU (1 x GPU)	CheXpert	88.09	78.55	16
Data parallel (4 x GPUs)	CheXpert	88.36	79.25	14
DDP (4 x GPUs)	CheXpert	88.33	80.10	4
Data parallel (4 x GPUs)	MIMIC-CXR	90.27	80.97	181
DDP (4 x GPUs)	MIMIC-CXR	90.31	81.76	54

Implementation

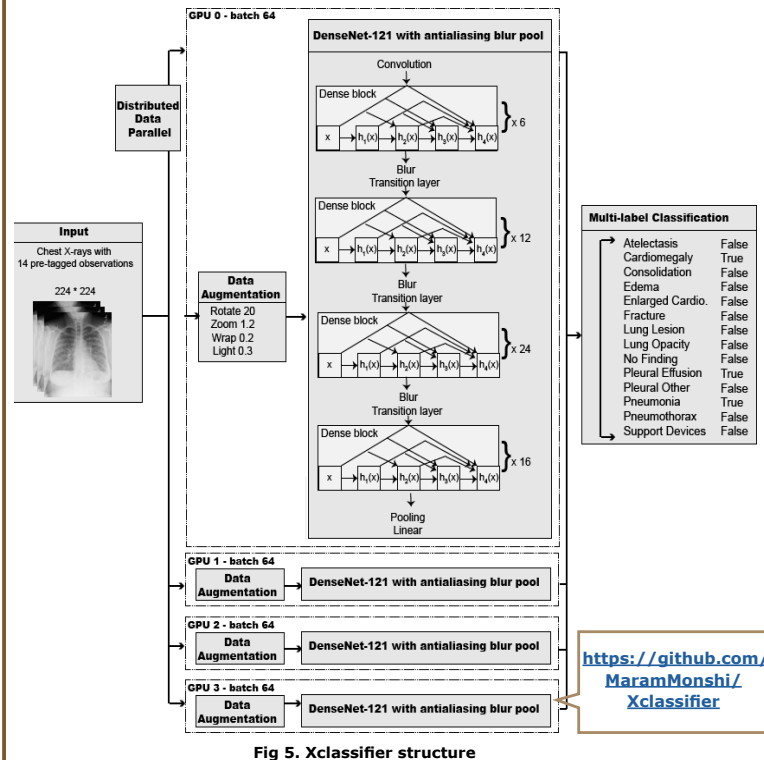


Fig 5. Xclassifier structure

Multi-label Classification

Atelectasis	False
Cardiomegaly	True
Consolidation	False
Edema	False
Enlarged Cardio.	False
Fracture	False
Lung Lesion	False
Lung Opacity	False
No Finding	False
Pleural Effusion	True
Pleural Other	False
Pneumonia	True
Pneumothorax	False
Support Devices	False

<https://github.com/MaramMonshi/Xclassifier>

Evaluation

Table 4: Image formats for chest x-rays and training performance

Chest x-ray format	Accuracy	AUC	Avg. time per epoch (min)
DICOM	89.40	80.02	111
JPEG	89.58	81.57	6

Digital Imaging and Communications in Medicine (DICOM) Joint Photographic Experts Group (JPEG)

Store medical imaging data Implement deep learning models



DICOM did not improve accuracy & took significantly more time to train than JPEG

Table 5: Comparing the Xclassifier with the benchmark

Multi-label classifier	Dataset	Accuracy	AUC
Latent-space self-ensemble (Gyawali et al., 2019)	CheXpert	—	66.97
CheXclusion (Seyyed-Kalantari et al., 2020)	CheXpert	—	80.50
Xclassifier	CheXpert	89.61	83.89
VSE-GCN (Hou et al., 2021)	MIMIC-CXR	—	72.10
CheXclusion (Seyyed-Kalantari et al., 2020)	MIMIC-CXR	—	83.40
Xclassifier	MIMIC-CXR	92.17	84.10

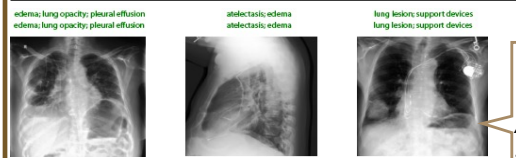


Fig 6. Correct output sample by the Xclassifier Model

Xclassifier improves multi-label classification performance by 0.70% AUC on MIMIC-CXR & by 3.39% AUC on CheXpert

Conclusion

- Contribution:
 - Propose Xclassifier, an efficient multi-label classifier that trains enhanced DenseNet-121 with blur pooling to detect 14 observations from CXRs
 - It accomplishes an ideal memory utilization, GPU computation, & high AUC on two large chest radiography datasets, MIMIC-CXR & CheXpert
- Future Work:
 - Investigate the use of DICOM in detecting diseases with small & complex structures to offer a greater degree of understanding of our initial findings
 - Concatenate patient data like age and gender to the flattened layer to improve prediction

References:

Bressen et al. (2020). Comparing different deep learning architectures for classification of chest radiographs.
 Chen et al. (2020). Label co-occurrence learning with graph convolutional networks for multi-label chest x-ray classification.
 Chowdhury et al. (2019). Semi-supervised learning by distilling and self-ensembling over stochastic latent space.
 Ho et al. (2021). Multi-label learning with visual attention, embedded knowledge graph for diagnosis of radiology imaging.
 Huang et al. (2017). Densely connected convolutional networks.
 Huang et al. (2019). Chestpert: A large chest radiograph dataset with uncertainty labels and expert comparison.
 Johnson et al. (2019). MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports.
 Li et al. (2019). PyTorch Distributed: Experiences on Accelerating Data Parallel Training.
 Mo & Cai (2019). Deep learning-based multi-label chest x-ray classification with entropy weighting loss.
 Monshi et al. (2021). CovidXpert: Optimizing Data Augmentation and CNN Hyperparameters for Improved COVID-19 Detection from CXR.
 Rajpurkar et al. (2017). Chest1501: Convolutional neural network to detect thoracic diseases from multi-view chest x-rays.
 Sabottke et al. (2020). ChestXpert: Radiologist-level pneumonia detection on chest x-rays with deep learning.
 Yarnall et al. (2020). The effect of image resolution on deep learning in radiology.
 Seyyed-Kalantari et al. (2020). CheXclusion: Feature gaps in deep chest X-ray classifiers.
 Wang et al. (2019). Tinet: Text-image embedding network for common thoracic disease classification and reporting in chest x-rays.
 Yao et al. (2017). Learning to diagnose from scratch by exploiting dependencies among labels.
 Yarnall (2020). X-Ray Classification Using Deep Learning and the MIMIC-CXR Dataset.
 Zhang (2019). Making convolutional networks shift invariant again.

Jupyter Notebook

CXRlabeler

What is CXRlabeler?

CXRlabeler is a deep learning labeler that takes raw radiology text as input and extracts 14 positive/negative CXR observations as its output. It utilizes the encoder learned from fine-tuning a language model on radiology reports in labeling these reports.

The implementation of CXRlabeler is available below, and more details about the data preparation and model comparison can be found on <https://github.com/MaramMonshi/CXRlabeler>.

Implementation:

- Python: 3.7.8
- PyTorch: 1.7.0
- fastai: 2.1.8
- GPU: 1 x NVIDIA Tesla V100 GPU
- Machine: n1highmem-8 (8 vCPUs, 52 GB memory)
- Platform: Linux-4.19.0-12-cloud-amd64-x86_64-with-debian-10.6

1 CXRlabeler Model

2 MIMIC-CXR Dataset

```
[45]: from fastai.basics import *
      from fastai.text.all import *
      import warnings
      warnings.filterwarnings('ignore')
      # Read in the train and test sets.
      path = Path('/home/jupyter/data/mimic-cxr')
      df_lm = pd.read_csv(path/"lm.csv")
      df_cl = pd.read_csv(path/"labels.csv")
      df_train = pd.read_csv(path/"train.csv")
      df_test = pd.read_csv(path/"test.csv")
```

```
[46]: df_lm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156790 entries, 0 to 156789
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   dicom_id    156790 non-null  object
1   reports     156790 non-null  object
dtypes: object(2)
memory usage: 2.4+ MB
```

```
[47]: df_cl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156790 entries, 0 to 156789
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  -
0   dicom_id    156790 non-null  object
1   reports     156790 non-null  object
2   Atelectasis 156790 non-null  int64
3   Cardiomegaly 156790 non-null  int64
```

```

4 Consolidation          156790 non-null int64
5 Edema                  156790 non-null int64
6 Enlarged Cardiomeastinum 156790 non-null int64
7 Fracture               156790 non-null int64
8 Lung Lesion            156790 non-null int64
9 Lung Opacity           156790 non-null int64
10 No Finding            156790 non-null int64
11 Pleural Effusion      156790 non-null int64
12 Pleural Other         156790 non-null int64
13 Pneumonia             156790 non-null int64
14 Pneumothorax          156790 non-null int64
15 Support Devices       156790 non-null int64
16 is_valid              156790 non-null bool
dtypes: bool(1), int64(14), object(2)
memory usage: 19.3+ MB

```

3 1. Language Model

```
[48]: df_text = pd.Series.append(df_train['reports'], df_test['reports'])
```

```
[49]: df_text = pd.DataFrame(df_text)
```

```
[50]: df_text.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 156790 entries, 0 to 3934
Data columns (total 1 columns):
#   Column   Non-Null Count  Dtype
---  -
0   reports  156790 non-null object
dtypes: object(1)
memory usage: 2.4+ MB

```

```
[51]: df_text.head()
```

```

[51]:          reports
0
No acute cardiopulmonary process.
1
No acute cardiopulmonary abnormality.
2
No acute intrathoracic process.
3 Focal consolidation at the left lung base, possibly representing aspiration
or\n pneumonia.\n \n Central vascular engorgement.
4
No evidence of acute cardiopulmonary process.

```

3.0.1 1.1 Data Block

```
[53]: bs_lm = (TextBlock.from_df(text_cols='reports', is_lm=True,
    ↪ tok_text_col='text'))
get_x = ColReader('text')
splitter = RandomSplitter(0.1, seed=42)
db_lm = DataBlock(blocks=bs_lm,
                  get_x=get_x,
                  splitter=splitter)
```

3.0.2 1.2 Data Loader

```
[54]: dl_lm = db_lm.dataloaders(df_text, bs=64)
```

<IPython.core.display.HTML object>

3.0.3 1.3 Training

```
[55]: learn_lm = language_model_learner(dl_lm, AWD_LSTM, pretrained=True,
    ↪ metrics=[accuracy, Perplexity()])
```

```
[56]: learn_lm.to_fp16()
learn_lm.fine_tune(10, 4e-3)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

3.0.4 1.4 Testing

```
[57]: N_WORDS = 40
N_SENTENCES = 2
```

```
[58]: TEXT = "Normal heart size. The right"
preds = [learn_lm.predict(TEXT, N_WORDS, temperature=0.75)
         for _ in range(N_SENTENCES)]
print("\n".join(preds))
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Normal heart size . The right hilar mass and calcified hilar lymph nodes
are consistent with sarcoidosis . No evidence of pneumonia . In comparison

with the study of ___ ,,, thethethe patientpatientpatient hashashas
takentakentaken aaa betterbetterbetter

inspirationinspirationinspiration ...
Normal heart size . The right PICC line is in adequate position . Comparison is
made to previous study from ___ atatat

4:534:534:53 a.m.a.m.a.m.

The endotracheal tube , feeding tube , and feeding tube are within the

```
[59]: TEXT = "Subtle patchy opacity along the"  
preds = [learn_lm.predict(TEXT, N_WORDS, temperature=0.75)  
         for _ in range(N_SENTENCES)]  
print("\n".join(preds))
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Subtle patchy opacity along the posterolateral left lower lung field
likely represents atelectasis , however , infection can not be completely
excluded . As compared to the previous radiograph , the lung volumes have
increased ,
causing increased crowding of pulmonary
Subtle patchy opacity along the right lateral chest wall which could
represent an early focus of pneumonia in the correct clinical setting . Right
lower lobe pneumonia . Follow up radiographs after ___

weekssweekssweekss areareare recommendedrecommendedrecommended tototo
showshowshow

```
[60]: TEXT = "Cardiomegaly is severe, unchanged. Pacemaker"  
preds = [learn_lm.predict(TEXT, N_WORDS, temperature=0.75)  
         for _ in range(N_SENTENCES)]  
print("\n".join(preds))
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Cardiomegaly is severe , unchanged . Pacemaker leads are in standard position with the leads terminating in the expected location of the right atrium and right ventricle . There is no pneumothorax . There is no pleural effusion or

pneumothorax .

Cardiomegaly is severe , unchanged . Pacemaker leads terminate in the right atrium and

right ventricle . There is no pneumothorax . Lungs are essentially clear .

No pleural effusion or pneumothorax is demonstrated . As compared to the previous

3.0.5 1.5 Saving

```
[61]: # save fine-tuned model for classification
learn_lm.save_encoder(path/'lm')
```

4 2. Multi-Label Classifier

```
[62]: # fix result
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
SEED = 42
seed_everything(SEED)
```

4.0.1 2.1 Data Block

```
[63]: labels = ["Atelectasis", "Cardiomegaly", "Consolidation",
               "Edema", "Enlarged Cardiomedastinum", "Fracture", "Lung Lesion",
               "Lung Opacity", "No Finding", "Pleural Effusion", "Pleural Other",
               "Pneumonia", "Pneumothorax", "Support Devices"]
```

```
[64]: dl_lm.seq_len
```

```
[64]: 72
```

```
dl_lm.vocab
```


[65]: dl_lm.vocab

[65]: ['xxunk', 'mild', 'congestion', 'can',
'xxpad', 'compared', 'catheter', 'sided',
'xxbos', 'as', 'since', 'ends',
'xxeos', 'study', 'could', 'most',
'xxfld', 'effusions', 'cm', 'view',
'xxrep', 'moderate', 'may', 'infection',
'xxwrep', 'comparison', 'by', 'severe',
'xxup', 'size', 'improved', 'aspiration',
'xxmaj', 'lobe', 'clear', 'within',
'.', 'evidence', 'mediastinal', 'picc',
'\n ', 'normal', 'stomach', 'if',
'the', 'radiograph', 'appearance', 'that',
'of', 'tip', 'from', 'over',
',', 'bilateral', 'have', 'carina',
'is', 'cardiac', 'focal', 'et',
'in', 'heart', 'svc', 'standard',
'no', 'cardiomegaly', 'mid', 'also',
'and', 'seen', 'bibasilar', 'devices',
'right', 'stable', 'lateral', 'excluded',
'to', 'previous', 'which', 'cardiopulmonary',
'left', 'for', 'opacification', 'support',
'pleural', 'opacities', 'ct', 'apical',
'with', 'upper', 'findings', 'central',
'3', 'silhouette', 'ap', 'cardiomediastinal',
'_', 'opacity', 'interstitial', 'jugular',
'there', 'not', 'again', 'well',
'are', 'change', 'process', 'basilar',
'lung', 'prior', 'more', 'fluid',
'pulmonary', 'vascular', 'consistent', 'dr',
'effusion', 'position', 'due', 'contours',
'a', 'volumes', 'an', 'relevant',
'atelectasis', 'but', 'slightly', 'place',
'tube', 'consolidation', 'minimal', ':',
'pneumothorax', '1', 'retrocardiac', 'venous',
'\n \n ', 'acute', 'was', 'some',
'edema', 'interval', 'clinical', 'enlargement',
'unchanged', 'new', 'above', 'overall',
'at', 'line', 'endotracheal', 'junction',
'chest', 'this', 'than', 'monitoring',
'or', 'been', 'large', 'persistent',
'-', 'likely', 'radiographs', 'reflect',
'pneumonia', '2', 'mediastinum', 'earlier',
'be', 'base', 'nasogastric', 'enlarged',
'has', 'patient', 'bases', 'setting',
'on', 'lungs', 'present', 'represent',
'lower', 'low', 'changes', 'otherwise',
'small', 'increased', 'parenchymal', 'decreased',

'without',	'possible',	'projecting',	'prominent',
'rib',	'appears',	'portable',	'advanced',
'however',	'subclavian',	'decrease',	'tracheostomy',
'level',	'limits',	'existing',	'pigtail',
'placement',	'approximately',	'elevation',	'time',
'were',	'reviewed',	'increasing',	'positioning',
'abnormality',	'definite',	'obtained',	'vein',
'appropriate',	'atrium',	'drainage',	'potentially',
'remains',	'superimposed',	'though',	'pacemaker',
'volume',	'pa',	'overt',	'p.m.',
'increase',	'known',	'after',	'fracture',
'areas',	'recent',	'resolution',	'its',
'basal',	'recommended',	'day',	'mass',
'substantial',	'similar',	'projects',	'costophrenic',
'region',	'demonstrated',	'into',	'resolved',
'it',	'further',	'remain',	'telephone',
'improvement',	'distal',	'top',	'wall',
'noted',	'larger',	'residual',	'feeding',
'little',	'slight',	'considered',	'possibility',
'now',	'port',	'middle',	'layering',
'other',	'infectious',	'less',	'terminating',
'internal',	'continued',	'minimally',	'pacer',
'although',	'persists',	'cavoatrial',	'extends',
'made',	'perihilar',	'difficult',	'followup',
'both',	'below',	'through',	'pressure',
'would',	'loss',	'exam',	'substantially',
'hilar',	'any',	'associated',	'lead',
'probably',	'borderline',	'emphysema',	'hemithorax',
'chronic',	'tubes',	'atelectatic',	'superior',
'aorta',	'terminates',	'discussed',	'out',
'removed',	'image',	'overload',	'aortic',
'disease',	'side',	'ij',	'wires',
'should',	'pre',	'collapse',	'suggesting',
'better',	'underlying',	'grossly',	'portion',
'patchy',	'diffuse',	'these',	'signs',
'worsening',	'extensive',	'previously',	'prominence',
'post',	'significant',	'diaphragm',	'linear',
'extent',	'adjacent',	'mildly',	'consolidations',
'air',	'unremarkable',	'exclude',	'upright',
'hemidiaphragm',	'area',	'engorgement',	'thickening',
'ng',	'worsened',	'multifocal',	'pneumothoraces',
'constant',	'appreciable',	'airspace',	''s'',
'concerning',	'greater',	'combination',	'least',
'examination',	'suggest',	'fractures',	'date',
'still',	'identified',	'a.m.',	'removal',
'thoracic',	'essentially',	'status',	'scarring',
'/',	'given',	'leads',	'apex',

'performed',	'presence',	'posterior',	'esophageal',
'compressive',	'chf',	'either',	'calcified',
'evaluation',	'early',	'apparent',	'somewhat',
'aeration',	'correlation',	'appreciated',	'obscured',
'multiple',	'back',	'abdomen',	'inspiration',
'structures',	'esophagus',	'when',	'hernia',
'developing',	'5',	'improving',	'failure',
'severity',	'possibly',	'trace',	'evaluated',
'course',	'current',	'asymmetric',	'persist',
'changed',	'nodules',	'ventricular',	'opacifications',
'abnormalities',	'including',	'helpful',	'limited',
'might',	'vessels',	'pulled',	'follow',
'passes',	'nodular',	'i',	'heterogeneous',
'4',	'infiltrate',	'pericardial',	'complete',
'vasculature',	'cath',	'artery',	'particularly',
'subcutaneous',	'concern',	'compatible',	'descending',
'visualized',	'related',	'newly',	'soft',
'amount',	'subtle',	'recently',	'marked',
'widening',	'zone',	'degree',	'intrathoracic',
'enteric',	'radiographic',	'completely',	'one',
'appear',	'component',	'along',	're',
'pic',	'following',	'repeat',	'shift',
'proximal',	'developed',	'bibasal',	'received',
'suggestive',	'bilaterally',	'tiny',	'device',
'loculated',	'parenchyma',	'placements',	'positioned',
'reflecting',	'sternotomy',	'smaller',	'finding',
'frontal',	'moderately',	'body',	'notably',
'very',	'suggests',	'midline',	'distended',
'markings',	'swan',	'assessment',	')',
'accompanied',	'ganz',	'density',	'several',
'elevated',	'free',	'technique',	'displaced',
'views',	'all',	'history',	'evident',
'probable',	'tortuosity',	'up',	'currently',
'sternal',	'correct',	'note',	'imaging',
'ventricle',	'expected',	'assess',	'film',
'relatively',	'lobes',	'tortuous',	'assessed',
'lines',	'shows',	'location',	'versus',
'postoperative',	'clinically',	'complications',	'anterior',
'development',	'single',	'represents',	'two',
'overlying',	'continues',	'contour',	'allowing',
'subsequent',	'same',	'paged',	'before',
'dobbhoff',	'angle',	'silhouettes',	'images',
'blunting',	'partially',	'pectoral',	'except',
';',	'progression',	'worse',	'carinal',
'recommend',	'widespread',	'spine',	'thorax',
'today',	'nodule',	'dedicated',	'phone',
'visible',	'only',	'differences',	'studies',

'additional',	'progressed',	'involving',	'extending',
'just',	'6',	'vertebral',	'tissue',
'drain',	'gas',	'confluent',	'differential',
'surfaces',	'ventilation',	'markedly',	'even',
'hemorrhage',	'old',	'remaining',	'observation',
'dual',	'treatment',	'referring',	'indistinctness',
'reflects',	'widened',	'supine',	'rounded',
'trachea',	'tension',	'near',	'demonstrates',
'neck',	'parts',	'partial',	'gastric',
'median',	'ge',	'results',	'margin',
'fully',	'available',	'adenopathy',	'does',
'representing',	'streaky',	'veins',	'close',
'diameter',	'main',	'so',	'detected',
'atrial',	'distribution',	'longer',	'conventional',
'pronounced',	'brachiocephalic',	'abdominal',	'field',
'lymphadenopathy',	'pneumomediastinum',	'alveolar',	'procedure',
'extubated',	'dilated',	'almost',	'calcification',
'cleared',	'dictation',	'addition',	'whether',
'hours',	'intubated',	'intact',	'ray',
'mm',	'border',	'subsegmental',	'weeks',
'hazy',	'physician',	'significantly',	'hila',
'expanded',	'consider',	'copd',	'imaged',
'surgical',	'such',	'fibrosis',	'notification',
'between',	'hemi',	'bronchial',	'appreciably',
'via',	'worrisome',	'communicated',	'predominantly',
'discovery',	'dense',	'rotation',	'show',
'described',	'short',	'redistribution',	'resolving',
'transvenous',	'collection',	'preceding',	'about',
'defined',	'diagnosis',	'sinus',	'concurrent',
'advised',	'crowding',	'secondary',	'alignment',
'suggested',	'x',	'hilus',	'review',
'presumed',	'infiltrates',	'malignancy',	'channel',
'much',	'part',	'decompensation',	'atypical',
'lesion',	'hyperinflated',	'congestive',	'days',
'clips',	'specifically',	'elongation',	'presumably',
'entirely',	'stent',	'ill',	'scan',
'please',	'non',	'metastatic',	'indeterminate',
'engorged',	'see',	'next',	'hypertension',
'placed',	'gastroesophageal',	'space',	'beyond',
'medial',	'cabg',	'peribronchial',	'(',
'taken',	'coiled',	'lies',	'bowel',
'absence',	'infrahilar',	'had',	'quadrant',
'aspect',	'regions',	'appropriately',	'healed',
'hiatal',	'high',	'bronchus',	'bronchovascular',
'surgery',	'clearly',	'raises',	'minor',
'valve',	'especially',	'optimal',	'throughout',
'withdrawn',	'appeared',	'fissure',	'positions',

'suspected',	'merely',	'balloon',	'lesions',
'collapsed',	'definitive',	'radiographically',	'report',
'distention',	'site',	'posteriorly',	'atelectases',
'correlate',	'inspiratory',	'arteries',	'ett',
'tissues',	'bronchiectasis',	'access',	'contrast',
'focus',	'their',	'advancement',	'extubation',
'vena',	'accentuate',	'doubt',	'despite',
'cava',	'satisfactory',	'yesterday',	'generator',
'calcifications',	'located',	'nearly',	'nonspecific',
'inlet',	'like',	'scoliosis',	'definition',
'laterally',	'clearing',	'catheters',	'severely',
'bony',	'transverse',	'supraclavicular',	'respectively',
'supervening',	'hyperinflation',	'provided',	'fat',
'second',	'suggestion',	'8',	'operative',
'evaluate',	'particular',	'medially',	'sheath',
'cervical',	'dobhoff',	'metastases',	'hilum',
'origin',	'subsequently',	'largely',	'third',
'compression',	'ribs',	'nipple',	'active',
'courses',	'series',	'masses',	'shoulder',
'term',	'replacement',	'terminate',	'ards',
'caused',	'leftward',	'wire',	'dilatation',
'peripheral',	'rotated',	'apparently',	'lingular',
'repositioned',	'fundus',	'background',	'hydropneumothorax',
'arch',	'zones',	'esophagogastric',	'quite',
'infusion',	'lymph',	'scanning',	'unclear',
'drains',	'bronchograms',	'document',	'warranted',
'replaced',	'oblique',	'ensure',	'diuresis',
'included',	'relate',	'perhaps',	'platelike',
'aside',	'where',	'tracheal',	'vague',
'rather',	'hemidiaphragms',	'hardware',	'reported',
'degenerative',	'pattern',	'past',	'sharply',
'convincing',	'inferior',	'hyperexpansion',	'indicative',
'symptoms',	'dependent',	'pump',	'relative',
'poor',	'10',	'indicated',	'raising',
'indwelling',	'minutes',	'accentuated',	'ascending',
'massive',	'need',	'obscuration',	'insertion',
'attention',	'defibrillator',	'sized',	'chin',
'caliber',	'projection',	'whose',	'etiology',
'thyroid',	'angles',	'loops',	'basis',
'remainder',	'progressive',	'situ',	'clavicular',
'pneumoperitoneum',	'lingula',	'overinflation',	'exaggerated',
'good',	'mitral',	'suspicious',	'dialysis',
'lucency',	'azygos',	'proper',	'intraperitoneal',
'semi',	'asymmetrical',	'radiation',	'required',
'orogastric',	'outside',	'axillary',	'end',
'coursing',	'cta',	'apices',	'4.5',
'intra',	'7',	'expansion',	'obvious',

```
'includes',      'nondistended',  'sensitive',    'torso',
'they',         'trauma',        'redemonstrated', 'hematoma',
'configuration', 'suspicion',    'aligned',      'node',
'repositioning', 'external',     'being',        'read',
'intubation',   '2.5',          'paratracheal', 'undergone',
'then',         'inserted',     'pneumonic',   'demonstrate',
'difference',   'diaphragmatic', 'potential',    'needs',
'midlung',     'towards',     'obscures',    'fibrotic',
'hemodialysis', 'films',       're-',         'decreasing',
'am',          'complication', 'generally',   'segment',
'reticular',   'interim',     'thoracentesis', 'ending',
'clavicle',    'vascularity', 'rest',        'first',
'inflated',    'explain',     'fields',      'far',
'asymmetry',  'aerated',     'resection',  ...]
```

```
[66]: dl_lm.max_vocab
```

```
[66]: 60000
```

```
[67]: bs_cl = (TextBlock.from_df('reports', seq_len=dl_lm.seq_len, vocab=dl_lm.vocab),
          MultiCategoryBlock(encoded=True, vocab=labels))
```

```
[68]: db_cl = DataBlock(blocks=bs_cl,
                        get_x=ColReader('text'),
                        get_y=ColReader(labels),
                        splitter=ColSplitter('is_valid'))
```

```
[69]: db_cl.summary(df_cl.iloc[:100])
```

Setting-up type transforms pipelines

Collecting items from

```
0  02aa804e-bde0afdd-112c0b34-7bc16630-4e384014
1  2a2277a9-b0ded155-c0de8eb9-c124d10e-82c5caab
2  68b5c4b1-227d0485-9cc38c3f-7b84ab51-4b472714
3  096052b7-d256dc40-453a102b-fa7d01c6-1b22c6b4
4  8959e402-2175d68d-edba5a6c-baab51c3-9359f700
..
95 325f2526-1ea870c1-06d8ff34-1b02764d-9e336cbc
96 38a433f3-1d000dff-a774352f-35c0d838-353e023f
97 4a25692b-e596ad27-5bc2eba3-e518093c-623f4d6a
98 0d24804d-197942ca-7f32a773-b93ba943-40022beb
99 a664e3c4-97f37598-e008ddb5-674d8b24-8a49114f
```

dicom_id \

reports \

```
0
No acute cardiopulmonary process.
1
```

No acute cardiopulmonary abnormality.

2

No acute intrathoracic process.

3

Focal consolidation at the left lung base, possibly representing aspiration or pneumonia. Central vascular engorgement.

4

No evidence of acute cardiopulmonary process.

..

...

95 Frontal and lateral views of the chest were obtained. Left basilar atelectasis is seen. There is left basilar and left mid lung atelectasis/scarring. Chain sutures are noted overlying the right upper-to-mid hemithorax. There is subtle focal patchy opacity projecting over the right lateral lower chest, which in the same location on the lateral view, appeared to be a linear opacity dating back to ___. Finding could represent atelectasis/scarring; however, on the current study, it appears more amorphous and a small focus of infection is not excluded. The cardiac and medi...

96

No pneumonia.

97 1. The left subclavian PICC line now has its tip in the distal SVC. Overall, cardiac and mediastinal contours are likely unchanged given differences in positioning. There is increased prominence of the pulmonary vasculature and indistinctness in the perihilar region consistent with interval appearance of mild interstitial and perihilar edema. No pleural effusions. No pneumothorax. Surgical chain sutures are again seen in the right upper lobe consistent with prior surgery. This is some fullness to the right suprahilar region which is unchanged and likely corresponds to ...

98

Fullness in the right lower paratracheal region of the mediastinum is comparable to the appearance in ___ when a chest CT scan showed no appreciable adenopathy in the mediastinum, instead a distended azygos vein. There was adenopathy in the adjacent right hilus, and the appearance of that structure is stable over these 3 examinations. Aside from small areas of linear scarring, lungs are clear. There is no edema or pneumonia and no appreciable pleural effusion. Heart size is normal.

99

As compared to the previous radiograph, the lung volumes have slightly decreased. There is minimal fluid overload in both the vascular and interstitial compartment. Normal size of the cardiac silhouette. Moderate tortuosity of the thoracic aorta. No pleural effusions. No pneumonia.

	Atelectasis	Cardiomegaly	Consolidation	Edema	\
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0

```

[100 rows x 17 columns]
Found 100 items
2 datasets of sizes 100,0
Setting up Pipeline: ColReader -- {'cols': 'text', 'pref': '', 'suff': '',
'label_delim': None} -> Tokenizer -> Numericalize
<IPython.core.display.HTML object>

Setting up Pipeline: ColReader -- {'cols': ['Atelectasis', 'Cardiomegaly',
'Consolidation', 'Edema', 'Enlarged Cardiomediatinum', 'Fracture', 'Lung
Lesion', 'Lung Opacity', 'No Finding', 'Pleural Effusion', 'Pleural Other',
'Pneumonia', 'Pneumothorax', 'Support Devices'], 'pref': '', 'suff': '',
'label_delim': None} -> EncodedMultiCategorize -- {'vocab': ['Atelectasis',
'Cardiomegaly', 'Consolidation', 'Edema', 'Enlarged Cardiomediatinum',
'Fracture', 'Lung Lesion', 'Lung Opacity', 'No Finding', 'Pleural Effusion',
'Pleural Other', 'Pneumonia', 'Pneumothorax', 'Support Devices'], 'sort': False,
'add_na': False}
Setting up after_item: Pipeline: ToTensor
Setting up before_batch: Pipeline: Pad_Chunk -- {'pad_idx': 1, 'pad_first':
True, 'seq_len': 72}
Setting up after_batch: Pipeline:

Building one batch
Applying item_tfms to the first sample:
  Pipeline: ToTensor
    starting from
      (TensorText([ 2,  8, 16, 81, 160, 120,  9]), TensorMultiCategory([0.,
0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]))
    applying ToTensor gives
      (TensorText([ 2,  8, 16, 81, 160, 120,  9]), TensorMultiCategory([0.,
0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]))

Adding the next 3 samples

Applying before_batch to the list of samples
  Pipeline: Pad_Chunk -- {'pad_idx': 1, 'pad_first': True, 'seq_len': 72}
    starting from
      [(TensorText([ 2,  8, 16, 81, 160, 120,  9]),
TensorMultiCategory([0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])),
(TensorText([ 2,  8, 16, 81, 160, 194,  9]), TensorMultiCategory([0., 0.,
0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])), (TensorText([ 2,  8, 16,
81, 535, 120,  9]), TensorMultiCategory([0., 0., 0., 0., 0., 0., 0., 0., 1.,
0., 0., 0., 0., 0.])), (TensorText of size 23, TensorMultiCategory([0., 0., 1.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]))]
    applying Pad_Chunk -- {'pad_idx': 1, 'pad_first': True, 'seq_len': 72} gives
      ((TensorText of size 23, TensorMultiCategory([0., 0., 0., 0., 0., 0., 0.,
0., 1., 0., 0., 0., 0., 0.])), (TensorText of size 23, TensorMultiCategory([0.,
0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])), (TensorText of size 23,

```



```
TensorMultiCategory([0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]),  
(TensorText of size 23, TensorMultiCategory([0., 0., 1., 0., 0., 0., 0., 0., 0.,  
0., 0., 0., 0., 0.])))
```

Collating items in a batch

No batch_tfms to apply

```
[70]: db_cl.splitter(df_cl)
```

```
[70]: ((#152855) [0,1,2,3,4,5,6,7,8,9...],  
      (#3935) [128,129,130,131,132,133,134,135,279,280...])
```

4.0.2 2.2 Data Loader

```
[71]: dl_cl = db_cl.dataloaders(df_cl)
```

<IPython.core.display.HTML object>

```
[72]: dl_cl.show_batch()
```

<IPython.core.display.HTML object>

4.0.3 2.3 Training

```
[74]: loss_func = BCEWithLogitsLossFlat(thresh=0.8)  
metrics = [partial(accuracy_multi, thresh=0.8),  
           F1ScoreMulti(average='macro'),  
           PrecisionMulti(average='macro'),  
           RecallMulti(average='macro'),  
           RocAucMulti(average='macro')]
```

```
[75]: learn_cl = text_classifier_learner(dl_cl, AWD_LSTM, metrics=metrics,   
    ↪ loss_func=loss_func)
```

```
[76]: learn_cl.load_encoder(path/'lm');
```

```
[77]: learn_cl.to_fp16()  
learn_cl.fine_tune(10)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

4.0.4 2.4 Testing

```
[78]: learn_cl.validate()
```

<IPython.core.display.HTML object>

```
[78]: (#6) [0.02500656060874462,0.9922853708267212,0.9617246278281069,0.959188927317167,0.9644917933863287,0.9980315733549691]
```

```
[79]: # labels = ["Atelectasis", "Cardiomegaly", "Consolidation",  
#           "Edema", "Enlarged Cardiomeastinum", "Fracture", "Lung  
↳Lesion",  
#           "Lung Opacity", "No Finding", "Pleural Efusion", "Pleural  
↳Other",  
#           "Pneumonia", "Pneumothorax", "Support Devices"]
```

```
[80]: # correct: 1, 0, 0, 1, 0, 0, 0 ,0, 0, 1, 0, 0, 0, 0  
learn_cl.predict("""Compared to chest radiographs since ____, most recently ____.  
  
Large right and moderate left pleural effusions and severe bibasilar  
atelectasis are unchanged. Cardiac silhouette is obscured. No pneumothorax.  
Pulmonary edema is mild, obscured radiographically by overlying abnormalities.↳  
↳""")
```

<IPython.core.display.HTML object>

```
[80]: ((#3) ['Atelectasis','Edema','Pleural Effusion'],  
tensor([ True, False, False, True, False, False, False, False, True,  
        False, False, False, False]),  
tensor([9.9958e-01, 2.4344e-03, 1.8244e-03, 9.7096e-01, 1.6865e-04, 5.1527e-04,  
        3.4268e-05, 3.1259e-04, 7.6621e-05, 9.9919e-01, 3.3475e-05, 1.8387e-03,  
        7.3686e-05, 1.5844e-04]))
```

```
[81]: # correct: 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1  
learn_cl.predict("""New mild pulmonary vascular congestion with mild to↳  
↳moderate interstitial  
pulmonary edema and increased mild cardiomegaly. No focal consolidation.""")
```

<IPython.core.display.HTML object>

```
[81]: ((#2) ['Cardiomegaly','Edema'],  
tensor([False, True, False, True, False, False, False, False, False, False,  
        False, False, False, False]),  
tensor([6.0708e-04, 1.0000e+00, 4.1335e-05, 9.9996e-01, 8.2679e-06, 1.5089e-05,  
        3.1137e-06, 6.7193e-04, 9.9462e-11, 1.2831e-04, 1.2030e-05, 2.5714e-04,  
        2.0785e-05, 5.3788e-04]))
```

```
[82]: # correct: 0, 1, 0, 1, 0, 0 ..
learn_cl.predict("""The heart remains enlarged. There is opacity along the
↳medial left
hemidiaphragm, which is known to correspond to a Bochdalek's hernia containing
portion of the left kidney. The right Bochdalek hernia is not as well
visualized on today's examination. There is patchy bibasilar opacity with
likely associated layering effusions. These findings could reflect
compressive atelectasis, although aspiration pneumonia should also be
considered. There is also an asymmetric airspace process involving the left
apex, which likely is not significantly changed dating all the way back to
___ and therefore would favor a benign process. No pulmonary edema. No
pneumothoraces. Mediastinal contours are stable.""")
```

<IPython.core.display.HTML object>

```
[82]: ((#4) ['Cardiomegaly', 'Lung Opacity', 'Pleural Effusion', 'Pneumonia'],
tensor([False, True, False, False, False, False, False, True, False, True,
False, True, False, False]),
tensor([4.8858e-02, 9.9418e-01, 1.6403e-02, 5.9803e-03, 2.1085e-02, 3.2504e-04,
5.2609e-03, 9.9921e-01, 5.0251e-05, 9.9440e-01, 3.3015e-04, 9.5615e-01,
4.9245e-03, 1.1774e-04]))
```

```
[83]: # correct: 1, 0, 0 ... , 1, 1
learn_cl.predict("""Comparison to the most recent preceding radiograph, there
↳is a
slight reaccumulation of fluid in the right pleural space. Two chest tubes
are noted in that space. A tiny apical pneumothorax is present. Right
atelectasis is also present. The left lung is essentially clear. Cardiac
size is normal.""")
```

<IPython.core.display.HTML object>

```
[83]: ((#3) ['Atelectasis', 'Pneumothorax', 'Support Devices'],
tensor([ True, False, False, False, False, False, False, False, False, False,
False, False, True, True]),
tensor([9.9842e-01, 1.2411e-05, 1.0868e-05, 1.0071e-04, 7.0141e-04, 1.5477e-04,
2.7752e-05, 4.7851e-06, 1.5978e-03, 3.0043e-03, 5.1739e-06, 8.4811e-05,
9.9654e-01, 9.9517e-01]))
```

```
[84]: # correct: 0, 0, 0, lung lesion, .., 1,0
learn_cl.predict("""1. Slowly growing peripheral right upper lobe lung nodule
↳is concerning for
primary lung adenocarcinoma. Dedicated chest CT may be considered for more
accurate assessment as well as to evaluate for possible right hilar lymph node
enlargement warranted clinically.
2. Low lung volumes limit assessment of the lung bases for pneumonia. Given
```

```
clinical suspicion for this entity, this could be further evaluated with repeat chest radiograph with improved inspiratory level. Dr. ___ was paged with these results at 8:15 a.m. on ___, at the time of discovery."""
```

<IPython.core.display.HTML object>

```
[84]: ((#2) ['Lung Lesion', 'Pneumonia'],  
       tensor([False, False, False, False, False, False,  True, False, False, False,  
              False,  True, False, False]),  
       tensor([1.7961e-03, 2.5072e-05, 2.9009e-03, 4.1085e-04, 1.9647e-03, 1.1412e-04,  
              9.8138e-01, 2.4845e-02, 1.6832e-05, 9.9153e-05, 4.1501e-03, 9.8510e-01,  
              4.7470e-04, 2.8797e-06]))
```

```
[85]: # correct: 0, 0, 0, fracture,0,1, ..., 0,0  
learn_cl.predict("""Subtle opacity projecting over the lateral right mid lung  
↳may be due to  
overlap of structures, but underlying pulmonary opacity or even rib fracture  
is not excluded. Findings could be further assessed with shallow oblique  
radiographs or chest CT. No displaced rib fracture definitively identified.↳  
↳However, if clinical  
concern persists, dedicated rib series or chest CT is more sensitive.""")
```

<IPython.core.display.HTML object>

```
[85]: ((#2) ['Fracture', 'Lung Opacity'],  
       tensor([False, False, False, False, False,  True, False,  True, False, False,  
              False, False, False, False]),  
       tensor([3.0774e-04, 3.1016e-04, 8.6147e-05, 4.0291e-04, 6.3868e-04, 9.4489e-01,  
              1.7892e-03, 9.9660e-01, 1.4426e-04, 1.4214e-03, 1.0987e-03, 3.9666e-04,  
              1.7892e-03, 2.8910e-04]))
```

4.0.5 2.5 Interpretation

```
[86]: interp = Interpretation.from_learner(learn_cl)
```

<IPython.core.display.HTML object>

```
[87]: interp.plot_top_losses(3)
```

<IPython.core.display.HTML object>

MultiViewModel

What is MultiViewModel?

MultiViewModel is a stage-wise model that is founded on a ResNet-50-based deep convolutional neural networks architecture to detect the presence and absence of 12 thorax diseases.

The implementation of MultiViewModel is available below, and more details about the data preparation and model comparison can be found on <https://github.com/MaramMonshi/MultiViewModel>.

Implementation:

- fastai: 1.0.61
- GPU: 4 x NVIDIA Tesla P4 GPUs
- Machine: n1-highmem-8 (8 vCPUs, 52 GB memory)

Multi-View Chest X-Rays

1.1 Using GPU

```
[1]: import torch
print(torch.cuda.is_available())
```

True

```
[2]: !nvidia-smi
```

Thu Jul 4 03:56:09 2019

```
+-----+
| NVIDIA-SMI 410.72          Driver Version: 410.72          CUDA Version: 10.0          |
+-----+-----+-----+-----+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|   0   Tesla P4             Off | 00000000:00:04.0 Off |                    |
| N/A   37C    P0     23W / 75W |  10MiB / 7611MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+
|   1   Tesla P4             Off | 00000000:00:05.0 Off |                    |
| N/A   37C    P0     23W / 75W |  10MiB / 7611MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+
|   2   Tesla P4             Off | 00000000:00:06.0 Off |                    |
| N/A   35C    P0     23W / 75W |  10MiB / 7611MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+
|   3   Tesla P4             Off | 00000000:00:07.0 Off |                    |
| N/A   38C    P0     23W / 75W |  10MiB / 7611MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+-----+-----+-----+-----+
| No running processes found                                     |
+-----+-----+-----+-----+-----+-----+
```

```
[110]: # learn.destroy() ## this Learner object self-destroyed - it still exists, but
↳no longer usable
torch.cuda.empty_cache()
```

1.2 Libraries

```
[1]: %reload_ext autoreload
%autoreload 2
%matplotlib inline
from fastai import *
from fastai.vision import *
from fastai.metrics import *
from fastai.callbacks import *
import warnings
warnings.filterwarnings('ignore')
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

1.3 Looking at Data

```
[2]: path = Path('/home/maram_m_monshi/data')
train_csv = pd.read_csv(path/'train.csv')
valid_csv = pd.read_csv(path/'valid.csv')
```

```
[5]: path.ls()
```

```
[5]: [PosixPath('/home/maram_m_monshi/data/train'),
PosixPath('/home/maram_m_monshi/data/train_p13'),
PosixPath('/home/maram_m_monshi/data/valid'),
PosixPath('/home/maram_m_monshi/data/train_p13.tar'),
PosixPath('/home/maram_m_monshi/data/valid.csv'),
PosixPath('/home/maram_m_monshi/data/train_p10'),
PosixPath('/home/maram_m_monshi/data/train_p12'),
PosixPath('/home/maram_m_monshi/data/models'),
PosixPath('/home/maram_m_monshi/data/train.csv'),
PosixPath('/home/maram_m_monshi/data/mimic-cxr')]
```

```
[3]: train_csv['Patient_Id'] = [Path(s).parents[0].parent.name for s in train_csv.
↳path]
valid_csv['Patient_Id'] = [Path(s).parents[0].parent.name for s in valid_csv.
↳path]
```

```
[6]: train_csv.shape, valid_csv.shape
```

```
[6]: ((37739, 16), (2732, 16))
```

```
[7]: df = pd.read_csv(path/'train.csv')
df.head()
```

```
[7]:
```

	path	view	No Finding	\
0	train/p11000011/s01/view1_frontal.jpg	frontal	NaN	
1	train/p11000183/s01/view1_frontal.jpg	frontal	NaN	
2	train/p11000183/s02/view1_frontal.jpg	frontal	NaN	
3	train/p11000183/s03/view1_frontal.jpg	frontal	NaN	
4	train/p11000183/s03/view2_frontal.jpg	frontal	NaN	

	Enlarged Cardiome-diastinum	Cardiomegaly	Airspace Opacity	Lung Lesion	\
0	NaN	NaN	NaN	NaN	
1	NaN	NaN	1.0	NaN	
2	NaN	NaN	1.0	NaN	
3	NaN	NaN	1.0	NaN	
4	NaN	NaN	1.0	NaN	

	Edema	Consolidation	Pneumonia	Atelectasis	Pneumothorax	\
0	NaN	0.0	1.0	NaN	NaN	
1	NaN	1.0	1.0	1.0	NaN	
2	NaN	NaN	NaN	1.0	NaN	
3	1.0	NaN	1.0	-1.0	NaN	
4	1.0	NaN	1.0	-1.0	NaN	

	Pleural Effusion	Pleural Other	Fracture	Support Devices
0	NaN	NaN	NaN	NaN
1	1.0	NaN	NaN	1.0
2	NaN	NaN	NaN	NaN
3	1.0	NaN	NaN	NaN
4	1.0	NaN	NaN	NaN

```
[8]: train_csv.path[:3]
```

```
[8]: 0 train/p11000011/s01/view1_frontal.jpg
1 train/p11000183/s01/view1_frontal.jpg
2 train/p11000183/s02/view1_frontal.jpg
Name: path, dtype: object
```

```
[9]: np.sum(train_csv['No Finding']) / len(train_csv)
```

```
[9]: 0.3705980550624023
```


1.4 Pre-Processing Data

```
[4]: tfms = get_transforms(True, False, max_rotate=None, max_zoom=0., max_lighting=0.
↳3,
                                max_warp=0, p_affine=0.5, p_lighting=0.5, xtra_tfms=[])
bs = 64; size = 224;
tfms
```

```
[4]: ([RandTransform(tfm=TfmCrop (crop_pad), kwargs={'row_pct': (0, 1), 'col_pct':
(0, 1), 'padding_mode': 'reflection'}, p=1.0, resolved={}, do_run=True,
is_random=True, use_on_y=True),
      RandTransform(tfm=TfmPixel (flip_lr), kwargs={}, p=0.5, resolved={},
do_run=True, is_random=True, use_on_y=True),
      RandTransform(tfm=TfmLighting (brightness), kwargs={'change': (0.35, 0.65)},
p=0.5, resolved={}, do_run=True, is_random=True, use_on_y=True),
      RandTransform(tfm=TfmLighting (contrast), kwargs={'scale': (0.7,
1.4285714285714286)}, p=0.5, resolved={}, do_run=True, is_random=True,
use_on_y=True)],
      [RandTransform(tfm=TfmCrop (crop_pad), kwargs={}, p=1.0, resolved={},
do_run=True, is_random=True, use_on_y=True)])
```

1.5 Evaluation (metrix for 2 classes)

```
[5]: from sklearn.metrics import roc_auc_score

class AUC(Callback):
    "AUC score"
    def __init__(self):
        pass

    def on_epoch_begin(self, **kwargs):
        self.outputs = []
        self.targets = []

    def on_batch_end(self, last_output, last_target, **kwargs):
        "expects binary output with data.c=2 "
        self.outputs += list(to_np(last_output)[: , 1])
        self.targets += list(to_np(last_target))

    def on_epoch_end(self, last_metrics, **kwargs):
        return {'last_metrics': last_metrics + [roc_auc_score(self.targets,
↳self.outputs)]}
auc = AUC()
```

2 Enlarged_Cardiomediastinum

```
[6]: enlarged_cardiomediastinum = 'Enlarged Cardiomediastinum'
train_enlarged_cardiomediastinum_csv = train_csv[['path',
↳enlarged_cardiomediastinum]].fillna(0).reset_index(drop=True)
# U-ignore: ignores uncertain predictions
train_enlarged_cardiomediastinum_csv =
↳train_enlarged_cardiomediastinum_csv[train_enlarged_cardiomediastinum_csv[en
larged_cardiomediastinum] != -1].reset_index(drop=True)
train_enlarged_cardiomediastinum_csv[enlarged_cardiomediastinum] =
↳train_enlarged_cardiomediastinum_csv[enlarged_cardiomediastinum].astype(int)
print(enlarged_cardiomediastinum)
print(train_enlarged_cardiomediastinum_csv['Enlarged Cardiomediastinum'].
↳value_counts(dropna=False))
train_enlarged_cardiomediastinum_csv[enlarged_cardiomediastinum].
↳value_counts(True)
```

```
Enlarged Cardiomediastinum
0    35367
1     1019
Name: Enlarged Cardiomediastinum, dtype: int64
```

```
[6]: 0    0.971995
1    0.028005
Name: Enlarged Cardiomediastinum, dtype: float64
```

```
[7]: itemlist = ImageList.from_df(df=train_enlarged_cardiomediastinum_csv,
↳path=path, folder='.', suffix='')
itemlists = itemlist.split_by_rand_pct(0.1)
data = (itemlists.label_from_df()
↳.transform(tfms, size=size)
↳.databunch(bs=bs)
↳.normalize(imagenet_stats))
itemlist, itemlists, data
```

```
[7]: (ImageList (36386 items)
Image (3, 3056, 2544),Image (3, 2544, 3056),Image (3, 2544, 3056),Image (3,
3056, 2544),Image (3, 3056, 2544)
Path: /home/maram_m_monshi/data, LabelLists;

Train: LabelList (32748 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;
```

```

Valid: LabelList (3638 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;

Test: None, ImageDataBunch;

Train: LabelList (32748 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;

Valid: LabelList (3638 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;

Test: None)

```

```
[8]: data.classes
```

```
[8]: [0, 1]
```

```
[10]: learn = cnn_learner(data=data, base_arch=models.resnet50, metrics=[accuracy,
→auc])
learn.model = nn.DataParallel(learn.model)
```

```
[11]: learn.fit_one_cycle(3)
```

<IPython.core.display.HTML object>

```
[12]: learn.lr_find()
```

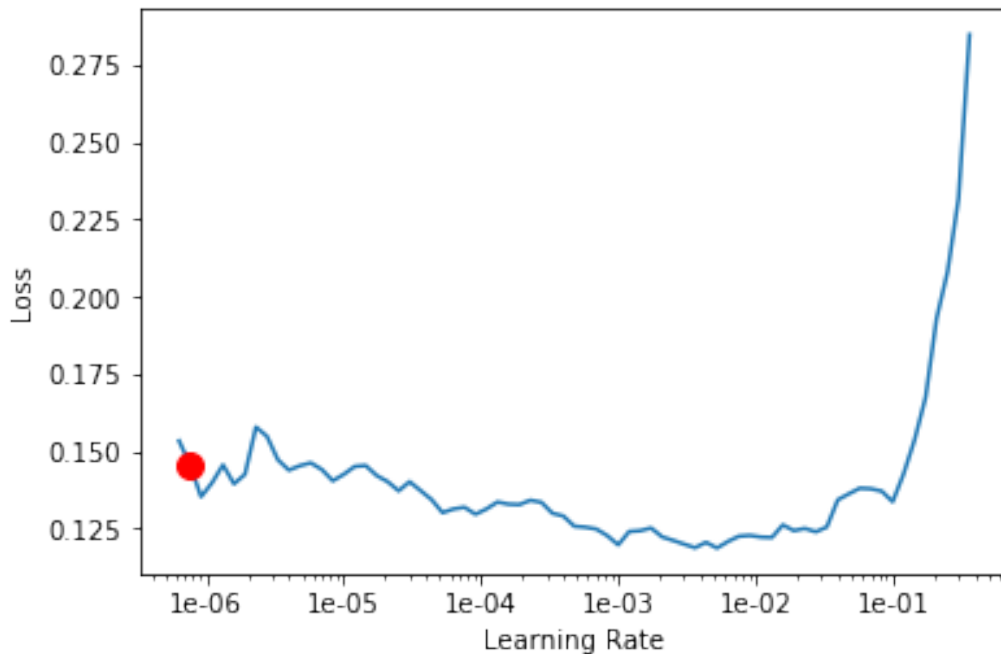
<IPython.core.display.HTML object>

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

```
[13]: learn.recorder.plot(suggestion=True)
```

Min numerical gradient: 7.59E-07

Min loss divided by 10: 5.25E-04



```
[14]: learn.save('enlarged_cardiomediatinum-stage-1')
```

```
[15]: interp = ClassificationInterpretation.from_learner(learn)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
```

```
[15]: True
```

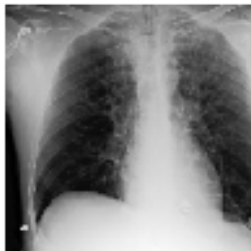
```
[16]: interp.plot_top_losses(9, figsize=(6,6))
```

prediction/actual/loss/probability

0/1 / 5.94 / 0.00



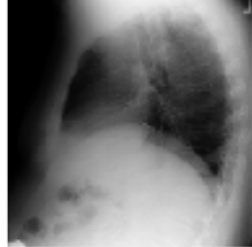
0/1 / 5.05 / 0.01



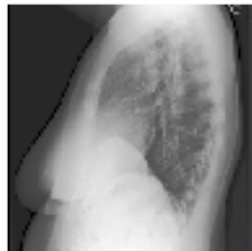
0/1 / 4.88 / 0.01



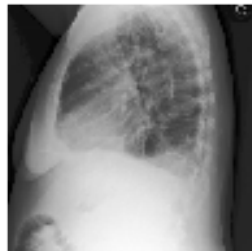
0/1 / 5.62 / 0.00



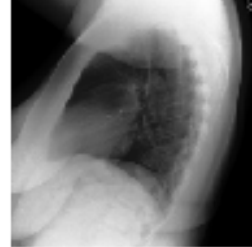
0/1 / 5.00 / 0.01



0/1 / 4.75 / 0.01



0/1 / 5.07 / 0.01



0/1 / 4.92 / 0.01



0/1 / 4.50 / 0.01



```
[18]: interp.most_confused(min_val=2)
```

```
[18]: [(1, 0, 116)]
```

```
[19]: learn.unfreeze()  
learn.fit_one_cycle(1)
```

<IPython.core.display.HTML object>

```
[20]: learn.load('enlarged_cardiomeastinum-stage-1')
```

```
[20]: Learner(data=ImageDataBunch;
```

Train: LabelList (32748 items)

x: ImageList

Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,

```
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;
```

```
Valid: LabelList (3638 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
0,0,0,0,0
Path: /home/maram_m_monshi/data;
```

```
Test: None, model=DataParallel(
  (module): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace)
      (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
      (4): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
          (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (relu): ReLU(inplace)
          (downsample): Sequential(
            (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
```

```

track_running_stats=True)
    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    (2): Bottleneck(
    (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    )
    (5): Sequential(
    (0): Bottleneck(
    (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    (downsample): Sequential(
    (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    )
    )

```

```

    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace)
    )
  )
)

```



```

(6): Sequential(
  (0): Bottleneck(
    (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    (downsample): Sequential(
      (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2),
bias=False)
      (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
  )
  (2): Bottleneck(
    (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),

```

```

bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    (3): Bottleneck(
    (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    (4): Bottleneck(
    (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    (5): Bottleneck(
    (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,

```

```

track_running_stats=True)
    (relu): ReLU(inplace)
  )
)
(7): Sequential(
  (0): Bottleneck(
    (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    (downsample): Sequential(
      (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2),
bias=False)
      (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
  )
  (2): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),

```

```

padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace)
    )
    )
    )
(1): Sequential(
  (0): AdaptiveConcatPool2d(
    (ap): AdaptiveAvgPool2d(output_size=1)
    (mp): AdaptiveMaxPool2d(output_size=1)
  )
  (1): Flatten()
  (2): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (3): Dropout(p=0.25)
  (4): Linear(in_features=4096, out_features=512, bias=True)
  (5): ReLU(inplace)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (7): Dropout(p=0.5)
  (8): Linear(in_features=512, out_features=2, bias=True)
)
)
), opt_func=functools.partial(<class 'torch.optim.adam.Adam'>, betas=(0.9,
0.99)), loss_func=FlattenedLoss of CrossEntropyLoss(), metrics=[<function
accuracy at 0x7f6bc37141e0>, AUC], true_wd=True, bn_wd=True, wd=0.01,
train_bn=True, path=PosixPath('/home/maram_m_monshi/data'), model_dir='models',
callback_fns=[functools.partial(<class 'fastai.basic_train.Recorder'>,
add_time=True, silent=False)], callbacks=[], layer_groups=[Sequential(
  (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (2): ReLU(inplace)
  (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (4): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (6): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,

```

```

track_running_stats=True)
  (8): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (10): ReLU(inplace)
  (11): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (13): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (14): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (15): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (16): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (17): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (19): ReLU(inplace)
  (20): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (21): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (22): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (23): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (24): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (25): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (26): ReLU(inplace)
  (27): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (28): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (29): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
  (30): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (31): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (32): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (33): ReLU(inplace)
  (34): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
  (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (36): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (37): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

```

```

(38): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
(39): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(40): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(42): ReLU(inplace)
(43): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(44): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(45): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
(46): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(47): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(48): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(49): ReLU(inplace)
(50): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(51): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(52): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
(53): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(54): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(55): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(56): ReLU(inplace)
), Sequential(
(0): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
(3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(4): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
(5): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(6): ReLU(inplace)
(7): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
(8): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(9): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(10): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,

```

```

track_running_stats=True)
  (11): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (13): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (14): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (15): ReLU(inplace)
  (16): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (17): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (18): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (19): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (20): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (21): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (22): ReLU(inplace)
  (23): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (24): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (25): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (26): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (27): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (28): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (29): ReLU(inplace)
  (30): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (31): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (32): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
  (33): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (34): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (35): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (36): ReLU(inplace)
  (37): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (38): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (39): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)

```

```

(40): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(41): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
(42): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(43): ReLU(inplace)
(44): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(45): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(46): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
(47): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(48): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(49): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(50): ReLU(inplace)
(51): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
(52): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(53): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(54): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(55): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
(56): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(57): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(58): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(59): ReLU(inplace)
(60): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(61): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(62): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
(63): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(64): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(65): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(66): ReLU(inplace)
), Sequential(
(0): AdaptiveAvgPool2d(output_size=1)
(1): AdaptiveMaxPool2d(output_size=1)
(2): Flatten()
(3): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True,

```



```
track_running_stats=True)
(4): Dropout(p=0.25)
(5): Linear(in_features=4096, out_features=512, bias=True)
(6): ReLU(inplace)
(7): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(8): Dropout(p=0.5)
(9): Linear(in_features=512, out_features=2, bias=True)
)], add_time=True, silent=None)
```

```
[21]: learn.lr_find()
```

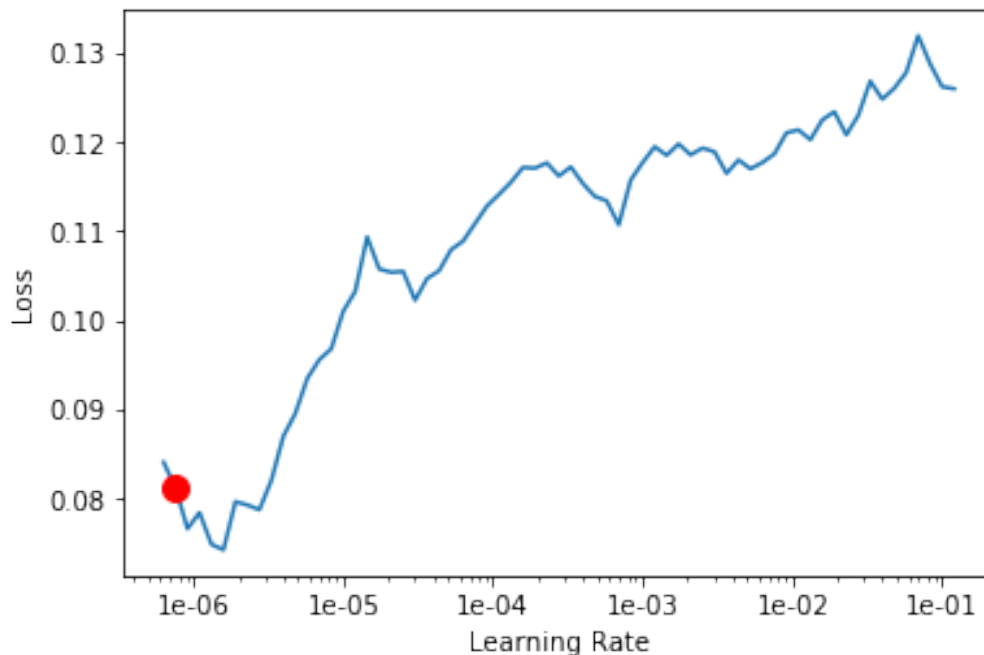
<IPython.core.display.HTML object>

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

```
[22]: learn.recorder.plot(suggestion=True)
```

Min numerical gradient: 7.59E-07

Min loss divided by 10: 1.58E-07



```
[23]: learn.unfreeze()
learn.fit_one_cycle(4, max_lr=slice(1e-6,1e-4))
```

<IPython.core.display.HTML object>

```
[24]: learn.save('enlarged_cardiomedastinum-stage-3')
```

9 Atelectasis

```
[30]: # create clean csv drop NAs
atelectasis = 'Atelectasis'
train_atelectasis_csv = train_csv[['path', atelectasis]].fillna(0).
↳reset_index(drop=True)
# U-ignore: ignores uncertain predictions
train_atelectasis_csv =
↳train_atelectasis_csv[train_atelectasis_csv[atelectasis] != -1].
↳reset_index(drop=True)
```

```
train_atelectasis_csv[atelectasis] = train_atelectasis_csv[atelectasis].
↳astype(int)
print(atelectasis)
print(train_atelectasis_csv['Atelectasis'].value_counts(dropna=False))
train_atelectasis_csv[atelectasis].value_counts(True)
```

```
Atelectasis
0    29876
1     6356
Name: Atelectasis, dtype: int64
```

```
[30]: 0    0.824575
1    0.175425
Name: Atelectasis, dtype: float64
```

```
[31]: itemlist = ImageList.from_df(df=train_atelectasis_csv, path=path, folder='.',
↳suffix='')
itemlists = itemlist.split_by_rand_pct(0.1)
data = (itemlists.label_from_df()
↳.transform(tfms, size=size)
↳.databunch(bs=bs)
↳.normalize(imagenet_stats))
```

```
itemlist, itemlists, data
```

```
[31]: (ImageList (36232 items)
Image (3, 3056, 2544),Image (3, 2544, 3056),Image (3, 2544, 3056),Image (3,
3056, 2544),Image (3, 2544, 3056)
```

```
Path: /home/maram_m_monshi/data, LabelLists;
```

```
Train: LabelList (32609 items)
```

```
x: ImageList
```

```
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
```

```
y: CategoryList
```

```
0,1,1,0,1
```

```
Path: /home/maram_m_monshi/data;
```

```
Valid: LabelList (3623 items)
```

```
x: ImageList
```

```
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
```

```
y: CategoryList
```

```
0,1,0,1,1
```

```
Path: /home/maram_m_monshi/data;
```

```
Test: None, ImageDataBunch;
```

```
Train: LabelList (32609 items)
```

```
x: ImageList
```

```
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
```

```
y: CategoryList
```

```
0,1,1,0,1
```

```
Path: /home/maram_m_monshi/data;
```

```
Valid: LabelList (3623 items)
```

```
x: ImageList
```

```
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
```

```
y: CategoryList
```

```
0,1,0,1,1
```

```
Path: /home/maram_m_monshi/data;
```

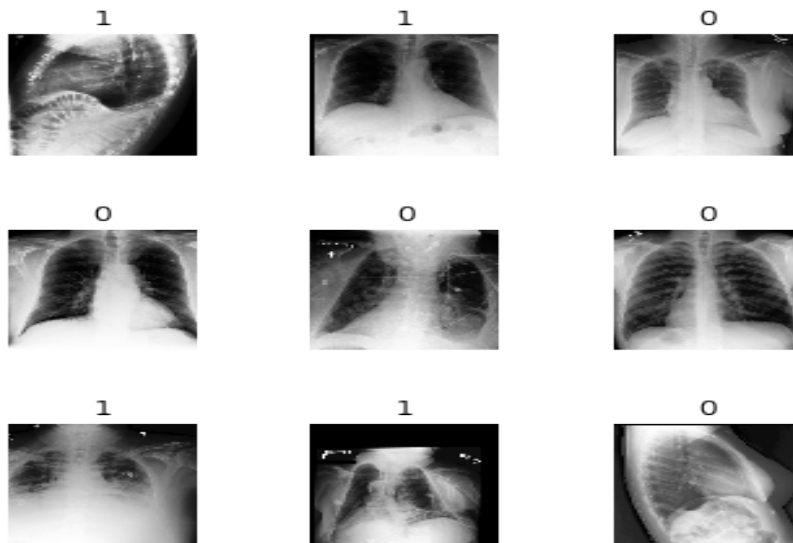
```
Test: None)
```

```
[32]: data.classes
```

```
[32]: [0, 1]
```

```
[33]: print(atelectasis)
data.show_batch(rows=3, figsize=(5,5))
```

Atelectasis



```
[34]: learn = cnn_learner(data=data, base_arch=models.resnet50, metrics=[accuracy, ↵
↵ auc])
learn.model = nn.DataParallel(learn.model)
```

```
[35]: learn.fit_one_cycle(3)
```

<IPython.core.display.HTML object>

```
[36]: learn.lr_find()
```

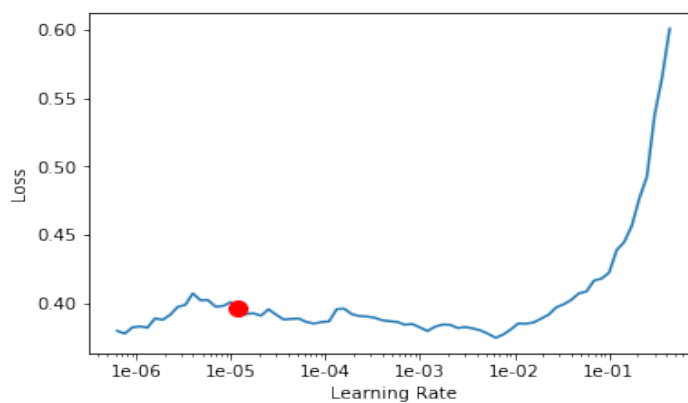
<IPython.core.display.HTML object>

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

```
[37]: learn.recorder.plot(suggestion=True)
```

Min numerical gradient: 1.20E-05

Min loss divided by 10: 6.31E-04

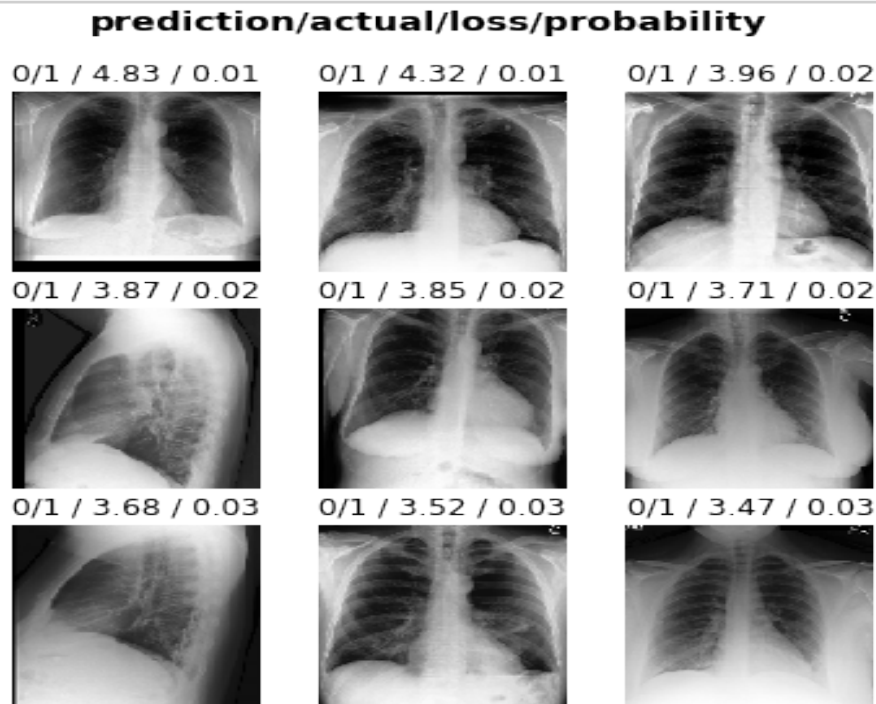


```
[38]: learn.save('atelectasis-stage-1')
```

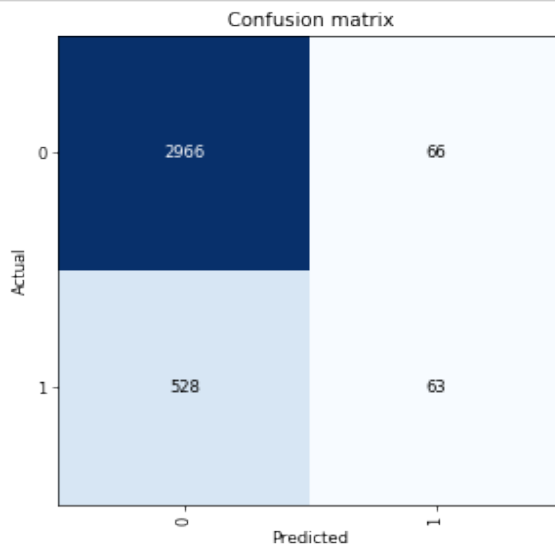
```
[39]: interp = ClassificationInterpretation.from_learner(learn)  
losses,idxs = interp.top_losses()  
len(data.valid_ds)==len(losses)==len(idxs)
```

[39]: True

```
[40]: interp.plot_top_losses(9, figsize=(6,6))
```



```
[41]: interp.plot_confusion_matrix(figsize=(5,5), dpi=60)
```



```
[42]: interp.most_confused(min_val=2)
```

```
[42]: [(1, 0, 528), (0, 1, 66)]
```

```
[43]: learn.unfreeze()  
learn.fit_one_cycle(1)
```

<IPython.core.display.HTML object>

```
[44]: learn.lr_find()
```

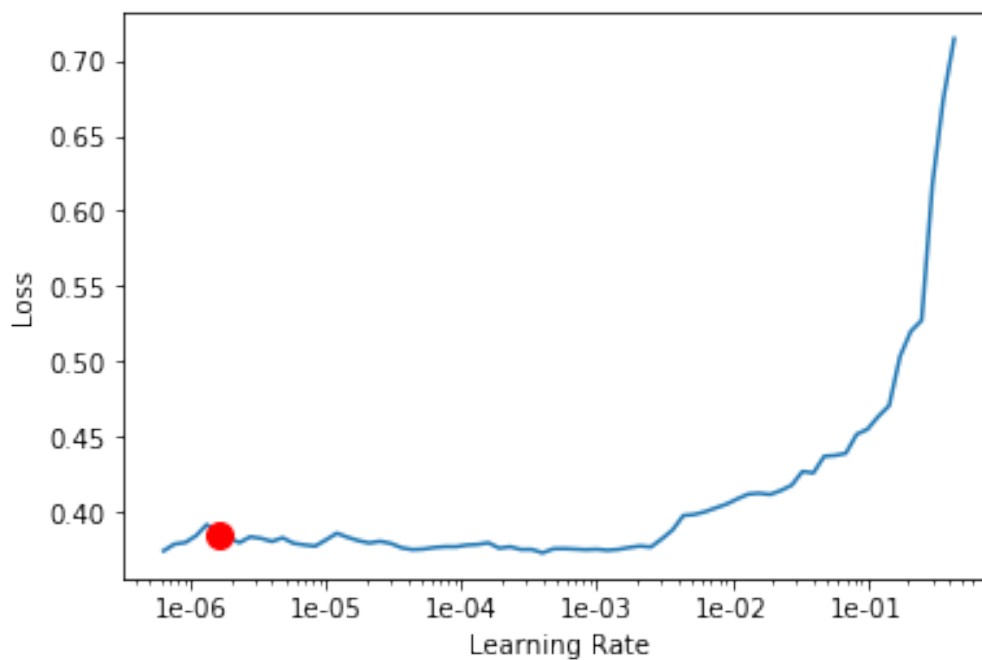
<IPython.core.display.HTML object>

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

```
[45]: learn.recorder.plot(suggestion=True)
```

Min numerical gradient: 1.58E-06

Min loss divided by 10: 3.98E-05



```
[46]: learn.unfreeze()  
learn.fit_one_cycle(4, max_lr=slice(1e-6, 1e-4))
```

<IPython.core.display.HTML object>

```
[47]: learn.save('atelectasis-stage-3')
```

Xclassifier

What is Xclassifier?

Xclassifier is an efficient multi-label classifier that trains an enhanced DenseNet-121 framework with blur pooling to detect 14 observations from a chest x-ray. It has ideal memory utilization and GPU computation, and a high AUC for two large chest radiography datasets, MIMIC-CXR, and CheXpert.

The implementation of Xclassifier is available below, and more details about the data preparation and model comparison can be found on <https://github.com/MaramMonshi/Xclassifier>.

Implementation:

- Python: 3.7.10
- PyTorch: 1.9.0
- fastai: 2.4
- GPU: 4 x NVIDIA Tesla V100 GPUs
- Machine: n1- highmem-32 (32 vCPUs, 208 GB memory)
- Platform: Linux-4.19.0-18-cloud-amd64-x86-64-with-debian-10.11

```
from fastai.vision.all import *
from timm import create_model
from fastai.vision.learner import _update_first_layer
from fastai.distributed import *
import os

def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True

SEED = 42
seed_everything(SEED)
os.chdir("/home/jupyter/data/mimic-cxr-jpg/")
torch.cuda.empty_cache()
import gc
gc.collect()
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('/home/jupyter/data/mimic-cxr-jpg/train-jpg.csv')
df = df.dropna(subset=['labels'])
df_test = pd.read_csv('/home/jupyter/data/mimic-cxr-jpg/test-jpg.csv')
df_test = df_test.dropna(subset=['labels'])

bs = 64
epoch = 50
metrics=[accuracy_multi,
          RocAucMulti(),
          PrecisionMulti(),
          RecallMulti(),
```



```

    F1ScoreMulti() ]

item_tfms=Resize(224, method='squish',
                 pad_mode='zeros', resamples=(2, 0))
batch_tfms=[*aug_transforms(mult=1.0, do_flip=False, flip_vert=False,
                            max_rotate=20.0, max_zoom=1.2,
                            max_lighting=0.3, max_warp=0.2,
                            p_affine=0.75, p_lighting=0.75,
                            xtra_tfms=None, size=None, mode='bilinear',
                            pad_mode='reflection', align_corners=True,
                            batch=False, min_scale=1.0),
            Normalize.from_stats(*imagenet_stats)]

dl = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
               get_x=ColReader('path'),
               get_y=ColReader('labels', label_delim=','),
               splitter=RandomSplitter(seed = SEED),
               item_tfms=item_tfms,
               batch_tfms=batch_tfms)

).dataloaders(df, bs=bs)

test_dl = dl.test_dl(df_test, with_labels=True)

def create_body(arch, n_in=3, pretrained=True, cut=None):
    "Cut off the body of the pretrained 'arch' as determined by 'cut'"
    model = arch(pretrained=pretrained)
    _update_first_layer(model, n_in, pretrained)
    if cut is None:
        ll = list(enumerate(model.children()))
        cut = next(i for i,o in reversed(ll) if has_pool_type(o))
    if isinstance(cut, int):
        return nn.Sequential(*list(model.children())[:cut])
    elif callable(cut): return cut(model)
    else: raise NamedError("cut must be either integer or a function")

```

```
def create_timm_body(arch:str, pretrained=True, cut=None, n_in=3):
    "Creates a body from a model in the 'timm' library."
    model = create_model(arch, pretrained=pretrained, num_classes=0,
                        global_pool='')
    _update_first_layer(model, n_in, pretrained)
    if cut is None:
        ll = list(enumerate(model.children()))
        cut = next(i for i,o in reversed(ll) if has_pool_type(o))
    if isinstance(cut, int):
        return nn.Sequential(*list(model.children())[:cut])
    elif callable(cut): return cut(model)
    else: raise NamedError("cut must be either integer or function")

body = create_timm_body('densenetblur121d', pretrained=True)
nf = num_features_model(body)
head = create_head(nf, dl.c, concat_pool=True)
net = nn.Sequential(body, head)
learn = Learner(dl, net, metrics=metrics)

with learn.distrib_ctx(sync_bn=False): learn.fine_tune(epoch);
    print (learn.validate(dl=test_dl))
```

COVIDcxr

What is COVIDcxr?

COVIDcxr is the dataset of 960 CXR images that we have proposed to introduce a balanced, unbiased, and complete COVID-19 CXR dataset.

A step-by-step guide to the creation of COVIDcxr dataset is available below, and more details about building a single neural network based on both images (CXRs) and tabular data (sex, age, and views) can be found on <https://github.com/MaramMonshi/CovidXrayNet/tree/main/Dataset>.

COVIDcxr-generate

November 5, 2021

1 Generate COVIDcxr Dataset

```
[1]: import numpy as np
import pandas as pd
import os
import random
from shutil import copyfile
import pydicom as dicom
import cv2
from fastai2.vision.all import *
```

1.1 1. COVID-19 Images

Download covid-19 image data collection from: <https://github.com/ieee8023/covid-chestxray-dataset>

```
[12]: # view csv file
covid19_csvpath = '/home/jupyter/covid-chestxray-dataset/metadata.csv'
dfcovid = pd.read_csv(covid19_csvpath)
dfcovid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 930 entries, 0 to 929
```

```
Data columns (total 29 columns):
```

#	Column	Non-Null Count	Dtype
0	patientid	930 non-null	object
1	offset	684 non-null	float64
2	sex	850 non-null	object
3	age	693 non-null	float64
4	finding	930 non-null	object
5	RT_PCR_positive	582 non-null	object
6	survival	358 non-null	object
7	intubated	243 non-null	object
8	intubation_present	246 non-null	object
9	went_icu	392 non-null	object
10	in_icu	331 non-null	object

11	needed_supplemental_02	88	non-null	object
12	extubated	37	non-null	object
13	temperature	74	non-null	float64
14	pO2_saturation	119	non-null	float64
15	leukocyte_count	16	non-null	float64
16	neutrophil_count	28	non-null	float64
17	lymphocyte_count	37	non-null	float64
18	view	930	non-null	object
19	modality	930	non-null	object
20	date	650	non-null	object
21	location	877	non-null	object
22	folder	930	non-null	object
23	filename	930	non-null	object
24	doi	382	non-null	object
25	url	930	non-null	object
26	license	685	non-null	object
27	clinical_notes	748	non-null	object
28	other_notes	424	non-null	object

dtypes: float64(7), object(22)

memory usage: 210.8+ KB

```
[13]: # clean the csv file
## drop CT images
dfcovid = dfcovid[dfcovid.modality != 'CT']
## drop other findings such as ARDS & Lateral View
dfcovid = dfcovid[dfcovid.finding == 'COVID-19']
dfcovid = dfcovid[dfcovid.view != 'L'] #23
## drop unused columns
dfcovid = dfcovid.drop(["offset", "RT_PCR_positive", "survival", "intubated",
↳ "intubation_present", "went_icu", "in_icu",
    "needed_supplemental_02", "extubated", "temperature",
    "pO2_saturation", "leukocyte_count", "neutrophil_count",
↳ "lymphocyte_count",
    "modality", "date", "location", "folder", "doi",
    "url", "license", "clinical_notes", "other_notes"], axis=1)
## drop NULL values
dfcovid = dfcovid.dropna(subset=['age'])
dfcovid = dfcovid.dropna(subset=['sex'])
dfcovid = dfcovid.dropna(subset=['view'])
## re-name columns
dfcovid = dfcovid.rename(columns={"filename": "path"})
## re-order columns
dfcovid = dfcovid[['path', 'finding', 'age', 'sex', 'view', 'patientid']]
## save dfcovid as a new clean csv
dfcovid.to_csv( "/home/jupyter/CovidXrayNet/covid.csv", index=False,
↳ encoding='utf-8-sig')
```

```
[14]: ## view covid.csv
dfcovid.info(), dfcovid.isnull().sum() ,dfcovid['finding'].value_counts(),
↳dfcovid['age'].value_counts(), dfcovid['sex'].value_counts(),
↳dfcovid['view'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 320 entries, 0 to 919
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   path        320 non-null    object
1   finding     320 non-null    object
2   age         320 non-null    float64
3   sex         320 non-null    object
4   view        320 non-null    object
5   patientid  320 non-null    object
dtypes: float64(1), object(5)
memory usage: 17.5+ KB
```

```
[14]: (None,
path          0
finding       0
age           0
sex           0
view          0
patientid    0
dtype: int64,
COVID-19     320
Name: finding, dtype: int64,
65.0      20
70.0      19
55.0      18
50.0      18
61.0      12
..
41.0      1
31.0      1
84.0      1
33.0      1
57.0      1
Name: age, Length: 62, dtype: int64,
M      213
F      107
Name: sex, dtype: int64,
AP      119
PA      117
AP Supine  84
```

```
Name: view, dtype: int64)
```

1.2 2. Pnumenia and Healthy Images

Download ChestX-ray14 dataset from: <https://www.kaggle.com/nih-chest-xrays/data>

```
[23]: # view chestxray14 csv file
chestxray14_csvpath = '/home/jupyter/Data_Entry_2017.csv'
dfpnumenia = pd.read_csv(chestxray14_csvpath)
dfpnumenia.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112120 entries, 0 to 112119
Data columns (total 12 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Image Index                               112120 non-null object
1   Finding Labels                            112120 non-null object
2   Follow-up #                               112120 non-null int64
3   Patient ID                               112120 non-null int64
4   Patient Age                              112120 non-null int64
5   Patient Gender                            112120 non-null object
6   View Position                             112120 non-null object
7   OriginalImage[Width                       112120 non-null int64
8   Height]
9   OriginalImagePixelSpacing[x               112120 non-null float64
10  y]
11  Unnamed: 11                               0 non-null      float64
dtypes: float64(3), int64(5), object(4)
memory usage: 10.3+ MB
```

```
[24]: # drop unused columns
dfpnumenia = dfpnumenia.drop(["Follow-up #", "OriginalImage[Width", "Height]",
                             "OriginalImagePixelSpacing[x","y]", "Unnamed: 11"], axis=1)
# arrange columns
dfpnumenia = dfpnumenia[['Image Index', 'Finding Labels', 'Patient Age',
↳ 'Patient Gender',
                             'View Position', 'Patient ID']]
# rename columns
dfpnumenia = dfpnumenia.rename(columns={"Image Index": "path", "Finding Labels":
↳ "finding",
                                     "Patient Age": "age", "Patient Gender": "sex",
                                     "View Position": "view", "Patient ID": "patientid"})
# 1. pnumenia
dfpnumenia = dfpnumenia[dfpnumenia.finding == 'Pneumonia']
dfpnumenia = dfpnumenia.drop(dfpnumenia.index[320:])
```

```
dfpnumenia.to_csv( "/home/jupyter/CovidXrayNet/pneumonia.csv", index=False,
↳encoding='utf-8-sig')
```

```
[26]: dfpnumenia.info(), dfpnumenia.isnull().sum(), dfpnumenia['finding'].
↳value_counts(), dfpnumenia['age'].value_counts(), dfpnumenia['sex'].
↳value_counts(), dfpnumenia['view'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 320 entries, 279 to 109877
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        320 non-null    object
1   finding     320 non-null    object
2   age         320 non-null    int64
3   sex         320 non-null    object
4   view        320 non-null    object
5   patientid   320 non-null    int64
dtypes: int64(2), object(4)
memory usage: 17.5+ KB
```

```
[26]: (None,
path          0
finding       0
age           0
sex           0
view          0
patientid     0
dtype: int64,
Pneumonia    320
Name: finding, dtype: int64,
33    12
46     9
44     9
50     9
63     8
..
75     1
77     1
79     1
82     1
3      1
Name: age, Length: 78, dtype: int64,
M    193
F    127
Name: sex, dtype: int64,
PA    176
```



```
AP      144
Name: view, dtype: int64)
```

```
[27]: # read chestxray14 csv file
chestxray14_csvpath = '/home/jupyter/Data_Entry_2017.csv'
dfhealthy = pd.read_csv(chestxray14_csvpath)
# drop unused columns
dfhealthy = dfhealthy.drop(["Follow-up #", "OriginalImage[Width", "Height]",
                           "OriginalImagePixelSpacing[x","y]", "Unnamed: 11"], axis=1)
# arrange columns
dfhealthy = dfhealthy[['Image Index', 'Finding Labels', 'Patient Age', 'Patient_
↳Gender',
                       'View Position', 'Patient ID']]
# rename columns
dfhealthy = dfhealthy.rename(columns={"Image Index": "path", "Finding Labels":
↳"finding",
                                     "Patient Age": "age", "Patient Gender": "sex",
                                     "View Position": "view", "Patient ID": "patientid"})
# 2. healthy
dfhealthy = dfhealthy[dfhealthy.finding == 'No Finding']
dfhealthy = dfhealthy.drop(dfhealthy.index[320:])
dfhealthy.to_csv( "/home/jupyter/CovidXrayNet/healthy.csv", index=False,
↳encoding='utf-8-sig')
```

```
[28]: dfhealthy.info(), dfhealthy.isnull().sum(), dfhealthy['finding'].
↳value_counts(), dfhealthy['age'].value_counts(), dfhealthy['sex'].
↳value_counts(), dfhealthy['view'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 320 entries, 3 to 787
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        320 non-null   object
1   finding     320 non-null   object
2   age         320 non-null   int64
3   sex         320 non-null   object
4   view        320 non-null   object
5   patientid   320 non-null   int64
dtypes: int64(2), object(4)
memory usage: 17.5+ KB
```

```
[28]: (None,
path      0
finding   0
age       0
sex       0)
```

```

view          0          54 8      57 4      85 2
patientid    0          56 7      78 4      80 1
dtype: int64,          58 7      87 3      84 1
No Finding    320       60 7      92 3      47 1
Name: finding, dtype: int64, 53 7      51 3      45 1
55 25         63 7      48 2      89 1
50 23         76 6      49 2      34 1
70 18         65 6      42 2      33 1
71 15         66 5      32 2      90 1
73 15         81 5      31 2      91 1
67 13         74 5      94 2      30 1
75 13         46 4      72 2      25 1
64 12         83 4      79 2      Name: age, dtype: int64,
69 12         59 4      82 2      M 162
61 10                                     F 158
77 9                                                   Name: sex, dtype: int64,
68 9                                                   PA 230
62 8                                                   AP 90
52 8                                                   Name: view, dtype: int64)
54 8

```

1.3 3. Combine Images

```

[2]: # view all csv files
path = Path('/home/jupyter/CovidXrayNet')
os.chdir("/home/jupyter/CovidXrayNet")
df_covid19 = pd.read_csv(path/'covid19.csv')
df_pneumonia = pd.read_csv(path/'pneumonia.csv')
df_healthy = pd.read_csv(path/'healthy.csv')

```

```

[9]: df_covid19.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 320 entries, 0 to 319
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        320 non-null    object
1   finding     320 non-null    object
2   age         320 non-null    float64
3   sex         320 non-null    object
4   view        320 non-null    object
5   patientid  320 non-null    object
dtypes: float64(1), object(5)
memory usage: 15.1+ KB

```

```

[10]: df_pneumonia.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 320 entries, 0 to 319
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        320 non-null    object
1   finding     320 non-null    object
2   age         320 non-null    int64
3   sex         320 non-null    object
4   view        320 non-null    object
5   patientid   320 non-null    int64
dtypes: int64(2), object(4)
memory usage: 15.1+ KB

```

```
[11]: df_healthy.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 320 entries, 0 to 319
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        320 non-null    object
1   finding     320 non-null    object
2   age         320 non-null    int64
3   sex         320 non-null    object
4   view        320 non-null    object
5   patientid   320 non-null    int64
dtypes: int64(2), object(4)
memory usage: 15.1+ KB

```

```
[14]: # Merge csv files vertically (on top of each other)
frames = [df_covid19, df_pneumonia, df_healthy]
df_covidcxr = pd.concat(frames)
df_covidcxr.to_csv("COVIDcxr.csv", index=False, encoding='utf-8-sig')
df_covidcxr.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 960 entries, 0 to 319
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   path        960 non-null    object
1   finding     960 non-null    object
2   age         960 non-null    float64
3   sex         960 non-null    object
4   view        960 non-null    object
5   patientid   960 non-null    object
dtypes: float64(1), object(5)

```

memory usage: 52.5+ KB

```
[15]: df_covidcxr.isnull().sum(), df_covidcxr['finding'].value_counts(),  
↳df_covidcxr['age'].value_counts(), df_covidcxr['sex'].value_counts(),  
↳df_covidcxr['view'].value_counts()
```

```
[15]: (path          0  
      finding       0  
      age           0  
      sex           0  
      view          0  
      patientid    0  
      dtype: int64,  
      COVID-19     320  
      No Finding   320  
      Pneumonia    320  
      Name: finding, dtype: int64,  
      55.0    51  
      50.0    50  
      70.0    37  
      65.0    32  
      73.0    29  
      ..  
      3.0     1  
      17.0    1  
      12.0    1  
      90.0    1  
      91.0    1  
      Name: age, Length: 91, dtype: int64,  
      M      568  
      F      392  
      Name: sex, dtype: int64,  
      PA      523  
      AP      353  
      AP Supine  84  
      Name: view, dtype: int64)
```

```
[16]: df_covidcxr['age'].describe()
```

```
[16]: count    960.000000  
      mean     54.669792  
      std      18.355041  
      min       3.000000  
      25%      44.000000  
      50%      56.000000  
      75%      69.000000  
      max      94.000000
```

Name: age, dtype: float64

```
[18]: ## move covid-19 images to one folder (if the path exist in csv)
dir_src = '/home/jupyter/covid-chestxray-dataset/images'
dir_dst = '/home/jupyter/covidcxr'
for fileName in df_covid19['path']:
    file_src = dir_src + "/" + fileName
    file_dst = dir_dst + "/" + fileName
    try:
        copyfile(file_src, file_dst)
    except IOError as e:
        print('Unable to copy file {} to {}'.format(file_src, file_dst))
    except:
        print('When try copy file {} to {}, unexpected error: {}'.format(file_src, file_dst, sys.exc_info()))
```

```
[26]: path = Path ('/home/jupyter/covidcxr')
num_files = len([f for f in os.listdir(path) if os.path.isfile(os.path.
↪join(path, f))])
num_files
```

[26]: 320

```
[ ]: # Run the following lines into the terminal to combine all nih-chestxray images
↪into one file
# mv /home/jupyter/images_001/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_002/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_003/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_004/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_005/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_006/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_007/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_008/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_009/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_010/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_011/images/* /home/jupyter/nih-chestxray
# mv /home/jupyter/images_012/images/* /home/jupyter/nih-chestxray
```

```
[43]: ## move pneumonia cxr to one folder (if the path exist in csv)
dir_src = '/home/jupyter/nih-chestxray'
dir_dst = '/home/jupyter/covidcxr'
for fileName in df_pneumonia['path']:
    file_src = dir_src + "/" + fileName
    file_dst = dir_dst + "/" + fileName
    try:
        copyfile(file_src, file_dst)
```

```
except IOError as e:
    print('Unable to copy file {} to {}'.format(file_src, file_dst))
except:
    print('When try copy file {} to {}, unexpected error: {}'.format(file_src, file_dst, sys.exc_info()))
```

```
[44]: path = Path ('/home/jupyter/covidcxr')
num_files = len([f for f in os.listdir(path)if os.path.isfile(os.path.
↪join(path, f))])
num_files
```

[44]: 640

```
[45]: ## move healthy cxr to one folder (if the path exist in csv)
for fileName in df_healthy['path']:
    file_src = dir_src + "/" + fileName
    file_dst = dir_dst + "/" + fileName
    try:
        copyfile(file_src, file_dst)
    except IOError as e:
        print('Unable to copy file {} to {}'.format(file_src, file_dst))
    except:
        print('When try copy file {} to {}, unexpected error: {}'.format(file_src, file_dst, sys.exc_info()))
```

```
[46]: path = Path ('/home/jupyter/covidcxr')
num_files = len([f for f in os.listdir(path)if os.path.isfile(os.path.
↪join(path, f))])
num_files
```

[46]: 960

```
[ ]:
```

CovidXrayNet

What is CovidXrayNet?

CovidXrayNet model improves the detection rate of coronavirus 2019 (COVID-19), from Chest X-Rays (CXRs) by means of optimizing the data augmentation pipeline and Convolutional Neural Network (CNN) hyperparameters.

The implementation of CovidXrayNet is available below, and more details about the data preparation and model comparison can be found on <https://github.com/MaramMonshi/CovidXrayNet>.

Implementation:

- Python: 3.7.6
- PyTorch: 1.6.0
- fastai: 2.0.16
- GPU: 1 x NVIDIA Tesla V100 GPU
- Machine: n1-highmem-8 (8 vCPUs, 52 GB memory)
- Platform: Linux-4.9.0-12-amd64-x86_64-with-debian-9.12

1 CovidXrayNet Model

1.1 COVIDx Dataset

```
[1]: from fastai.vision.all import *
import os.path
path = Path('/home/jupyter/covidx')
```

```
[2]: torch.cuda.empty_cache()
```

```
[3]: # fix result
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
SEED = 42
seed_everything(SEED)
```

```
[4]: df = pd.read_csv(path/'covidx.csv')
df
```

```
[4]:
```

	patientid	\
0		3
1		3
2		7
3		7
4		9
...		...
15491	2c917d3a-95cb-4c11-802c-f83e28cb37bc	
15492	3040d9d7-d895-453f-887c-616c10531960	
15493	c07f52df-d481-434f-84c1-04263926ac40	
15494	c109061a-d815-4cae-8343-9230d8024adf	
15495	c18d1138-ba74-4af5-af21-bdd4d2c96bb5	

	path	finding
0	SARS-10.1148rg.242035193-g04mr34g0-Fig8b-day5.jpeg	pneumonia
1	SARS-10.1148rg.242035193-g04mr34g0-Fig8c-day10.jpeg	pneumonia
2	SARS-10.1148rg.242035193-g04mr34g04a-Fig4a-day7.jpeg	pneumonia
3	SARS-10.1148rg.242035193-g04mr34g04b-Fig4b-day12.jpeg	pneumonia
4	SARS-10.1148rg.242035193-g04mr34g07a-Fig7a-day5.jpeg	pneumonia
...
15491	2c917d3a-95cb-4c11-802c-f83e28cb37bc.png	pneumonia
15492	3040d9d7-d895-453f-887c-616c10531960.png	pneumonia
15493	c07f52df-d481-434f-84c1-04263926ac40.png	pneumonia
15494	c109061a-d815-4cae-8343-9230d8024adf.png	pneumonia
15495	c18d1138-ba74-4af5-af21-bdd4d2c96bb5.png	pneumonia

	source	is_valid
0	cohen	False
1	cohen	False
2	cohen	False
3	cohen	False
4	cohen	False
...
15491	rsna	True
15492	rsna	True
15493	rsna	True
15494	rsna	True
15495	rsna	True

[15496 rows x 5 columns]

```
[5]: df['finding'].value_counts()
```

```
[5]: normal      8851
pneumonia      6056
COVID-19       589
Name: finding, dtype: int64
```

```
[6]: df['is_valid'].value_counts()
```

```
[6]: False      13917
True          1579
Name: is_valid, dtype: int64
```

1.2 Data Block

```
[7]: get_x=lambda x:path/"images"/f"{x[1]}"
      get_y=lambda x:x[2]
      splitter=ColSplitter('is_valid')
```

```
[8]: metrics=[accuracy,
              #RocAuc(average='macro', multi_class='ovr'),
              MatthewsCorrCoef(sample_weight=None),
              Precision(average='macro'),
              Recall(average='macro'),
              F1Score(average='macro')]
```

Data Augmentation Pipeline

```
[9]: item_tfms=Resize(480, method='squish', pad_mode='zeros', resamples=(2, 0))
      batch_tfms=[*aug_transforms(mult=1.0, do_flip=False, flip_vert=False,
                                  max_rotate=20.0, max_zoom=1.2, max_lighting=0.3,
                                  ↪max_warp=0.2,
                                  p_affine=0.75, p_lighting=0.75,
                                  xtra_tfms=None, size=None, mode='bilinear',
                                  ↪pad_mode='reflection',
                                  align_corners=True, batch=False, min_scale=1.0),
                  Normalize.from_stats(*imagenet_stats)]
```

```
[10]: db = DataBlock(blocks=(ImageBlock(cls=PILImageBW), CategoryBlock),
                     get_x=get_x,
                     get_y=get_y,
                     splitter=splitter,
                     item_tfms = item_tfms,
                     batch_tfms=batch_tfms)
```

```
[11]: db.item_tfms
```

```
[11]: (#2) [ToTensor:
encodes: (PILMask,object) -> encodes
(PILBase,object) -> encodes
decodes: ,Resize -- {'size': (480, 480), 'method': 'squish', 'pad_mode':
'zeros', 'resamples': (2, 0), 'p': 1.0}:
encodes: (Image,object) -> encodes
(TensorBBox,object) -> encodes
(TensorPoint,object) -> encodes
decodes: ]
```

```
[12]: db.batch_tfms
```

```
[12]: (#4) [IntToFloatTensor -- {'div': 255.0, 'div_mask': 1}]:
encodes: (TensorImage,object) -> encodes
(TensorMask,object) -> encodes
decodes: (TensorImage,object) -> decodes
,Warp -- {'magnitude': 0.2, 'p': 1.0, 'draw_x': None, 'draw_y': None, 'size':
None, 'mode': 'bilinear', 'pad_mode': 'reflection', 'batch': False,
'align_corners': True, 'mode_mask': 'nearest'}:
encodes: (TensorImage,object) -> encodes
(TensorMask,object) -> encodes
(TensorBBox,object) -> encodes
(TensorPoint,object) -> encodes
decodes: ,Brightness -- {'max_lighting': 0.3, 'p': 1.0, 'draw': None, 'batch':
False}:
encodes: (TensorImage,object) -> encodes
decodes: ,Normalize -- {'mean': tensor([[[[0.4850]],

[[0.4560]],

[[0.4060]]]], device='cuda:0'), 'std': tensor([[[[0.2290]],

[[0.2240]],

[[0.2250]]]], device='cuda:0'), 'axes': (0, 2, 3)}:
encodes: (TensorImage,object) -> encodes
decodes: (TensorImage,object) -> decodes
]
```

```
[13]: db.splitter(df)
```

```
[13]: ((#13917) [0,1,2,3,4,5,6,7,8,9...],
(#1579) [13917,13918,13919,13920,13921,13922,13923,13924,13925,13926...])
```

1.3 Data Loader

CNN Hyperparameters

```
[14]: from efficientnet_pytorch import EfficientNet
arch = EfficientNet.from_pretrained("efficientnet-b0")
bs = 32
epoch = 30
loss_func=LabelSmoothingCrossEntropyFlat(axis=-1, eps=0.2, reduction='mean',
↪flatten=True, floatify=False, is_2d=True)
```

Loaded pretrained weights for efficientnet-b0

```
[15]: dl = db.dataloaders(df, bs=bs)
```

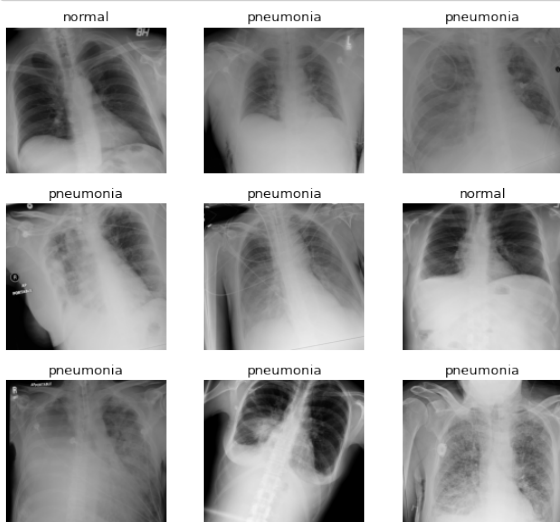
```
[16]: dl.after_item
```

```
[16]: Pipeline: Resize -- {'size': (480, 480), 'method': 'squish', 'pad_mode': 'zeros', 'resamples': (2, 0), 'p': 1.0} -> ToTensor
```

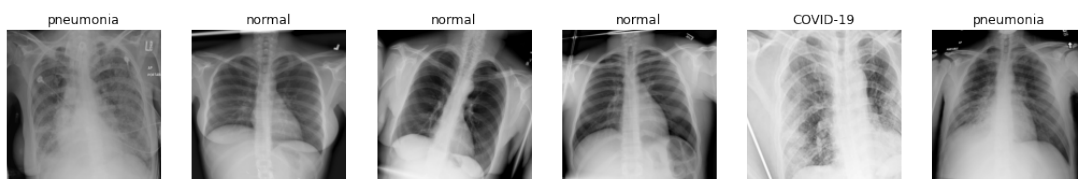
```
[17]: dl.after_batch
```

```
[17]: Pipeline: IntToFloatTensor -- {'div': 255.0, 'div_mask': 1} -> Warp -- {'magnitude': 0.2, 'p': 1.0, 'draw_x': None, 'draw_y': None, 'size': None, 'mode': 'bilinear', 'pad_mode': 'reflection', 'batch': False, 'align_corners': True, 'mode_mask': 'nearest'} -> Brightness -- {'max_lighting': 0.3, 'p': 1.0, 'draw': None, 'batch': False} -> Normalize -- {'mean': tensor([[[[0.4850]],  
  
[[0.4560]],  
  
[[0.4060]]]), device='cuda:0'), 'std': tensor([[[[0.2290]],  
  
[[0.2240]],  
  
[[0.2250]]]), device='cuda:0'), 'axes': (0, 2, 3)}
```

```
[18]: dl.show_batch()
```



```
[19]: dl.show_batch(max_n=6, nrows=1, ncols=6)
```



1.4 Training

```
[20]: learn = Learner(dl, model=arch, loss_func=loss_func, metrics=metrics)
```

```
[21]: learn.summary()
```

<IPython.core.display.HTML object>

```
[21]: EfficientNet (Input shape: ['32 x 3 x 480 x 480'])
```

```
=====
```

Layer (type)	Output Shape	Param #	Trainable
ZeroPad2d	32 x 3 x 481 x 481	0	False
BatchNorm2d	32 x 32 x 240 x 240	64	True
ZeroPad2d	32 x 32 x 242 x 242	0	False
BatchNorm2d	32 x 32 x 240 x 240	64	True
Identity	32 x 32 x 1 x 1	0	False
Identity	32 x 8 x 1 x 1	0	False
Identity	32 x 32 x 240 x 240	0	False
BatchNorm2d	32 x 16 x 240 x 240	32	True
MemoryEfficientSwish	32 x 8 x 1 x 1	0	False
Identity	32 x 16 x 240 x 240	0	False
BatchNorm2d	32 x 96 x 240 x 240	192	True
ZeroPad2d	32 x 96 x 241 x 241	0	False
BatchNorm2d	32 x 96 x 120 x 120	192	True
Identity	32 x 96 x 1 x 1	0	False
Identity	32 x 4 x 1 x 1	0	False
Identity	32 x 96 x 120 x 120	0	False
BatchNorm2d	32 x 24 x 120 x 120	48	True

```
=====
```

MemoryEfficientSwish	32 x 4 x 1 x 1	0	False
Identity	32 x 24 x 120 x 120	0	False
BatchNorm2d	32 x 144 x 120 x 12	288	True
ZeroPad2d	32 x 144 x 122 x 12	0	False
BatchNorm2d	32 x 144 x 120 x 12	288	True
Identity	32 x 144 x 1 x 1	0	False
Identity	32 x 6 x 1 x 1	0	False
Identity	32 x 144 x 120 x 12	0	False
BatchNorm2d	32 x 24 x 120 x 120	48	True
MemoryEfficientSwish	32 x 6 x 1 x 1	0	False
Identity	32 x 24 x 120 x 120	0	False
BatchNorm2d	32 x 144 x 120 x 12	288	True
ZeroPad2d	32 x 144 x 123 x 12	0	False
BatchNorm2d	32 x 144 x 60 x 60	288	True
Identity	32 x 144 x 1 x 1	0	False
Identity	32 x 6 x 1 x 1	0	False
Identity	32 x 144 x 60 x 60	0	False
BatchNorm2d	32 x 40 x 60 x 60	80	True
MemoryEfficientSwish	32 x 6 x 1 x 1	0	False
Identity	32 x 40 x 60 x 60	0	False
BatchNorm2d	32 x 240 x 60 x 60	480	True
ZeroPad2d	32 x 240 x 64 x 64	0	False
BatchNorm2d	32 x 240 x 60 x 60	480	True
Identity	32 x 240 x 1 x 1	0	False

Identity	32 x 10 x 1 x 1	0	False
Identity	32 x 240 x 60 x 60	0	False
BatchNorm2d	32 x 40 x 60 x 60	80	True
MemoryEfficientSwish	32 x 10 x 1 x 1	0	False
Identity	32 x 40 x 60 x 60	0	False
BatchNorm2d	32 x 240 x 60 x 60	480	True
ZeroPad2d	32 x 240 x 61 x 61	0	False
BatchNorm2d	32 x 240 x 30 x 30	480	True
Identity	32 x 240 x 1 x 1	0	False
Identity	32 x 10 x 1 x 1	0	False
Identity	32 x 240 x 30 x 30	0	False
BatchNorm2d	32 x 80 x 30 x 30	160	True
MemoryEfficientSwish	32 x 10 x 1 x 1	0	False
Identity	32 x 80 x 30 x 30	0	False
BatchNorm2d	32 x 480 x 30 x 30	960	True
ZeroPad2d	32 x 480 x 32 x 32	0	False
BatchNorm2d	32 x 480 x 30 x 30	960	True
Identity	32 x 480 x 1 x 1	0	False
Identity	32 x 20 x 1 x 1	0	False
Identity	32 x 480 x 30 x 30	0	False
BatchNorm2d	32 x 80 x 30 x 30	160	True
MemoryEfficientSwish	32 x 20 x 1 x 1	0	False
Identity	32 x 80 x 30 x 30	0	False

BatchNorm2d	32 x 480 x 30 x 30	960	True
ZeroPad2d	32 x 480 x 32 x 32	0	False
BatchNorm2d	32 x 480 x 30 x 30	960	True
Identity	32 x 480 x 1 x 1	0	False
Identity	32 x 20 x 1 x 1	0	False
Identity	32 x 480 x 30 x 30	0	False
BatchNorm2d	32 x 80 x 30 x 30	160	True
MemoryEfficientSwish	32 x 20 x 1 x 1	0	False
Identity	32 x 80 x 30 x 30	0	False
BatchNorm2d	32 x 480 x 30 x 30	960	True
ZeroPad2d	32 x 480 x 34 x 34	0	False
BatchNorm2d	32 x 480 x 30 x 30	960	True
Identity	32 x 480 x 1 x 1	0	False
Identity	32 x 20 x 1 x 1	0	False
Identity	32 x 480 x 30 x 30	0	False
BatchNorm2d	32 x 112 x 30 x 30	224	True
MemoryEfficientSwish	32 x 20 x 1 x 1	0	False
Identity	32 x 112 x 30 x 30	0	False
BatchNorm2d	32 x 672 x 30 x 30	1,344	True
ZeroPad2d	32 x 672 x 34 x 34	0	False
BatchNorm2d	32 x 672 x 30 x 30	1,344	True
Identity	32 x 672 x 1 x 1	0	False
Identity	32 x 28 x 1 x 1	0	False
Identity	32 x 672 x 30 x 30	0	False

BatchNorm2d	32 x 112 x 30 x 30	224	True
MemoryEfficientSwish	32 x 28 x 1 x 1	0	False
Identity	32 x 112 x 30 x 30	0	False
BatchNorm2d	32 x 672 x 30 x 30	1,344	True
ZeroPad2d	32 x 672 x 34 x 34	0	False
BatchNorm2d	32 x 672 x 30 x 30	1,344	True
Identity	32 x 672 x 1 x 1	0	False
Identity	32 x 28 x 1 x 1	0	False
Identity	32 x 672 x 30 x 30	0	False
BatchNorm2d	32 x 112 x 30 x 30	224	True
MemoryEfficientSwish	32 x 28 x 1 x 1	0	False
Identity	32 x 112 x 30 x 30	0	False
BatchNorm2d	32 x 672 x 30 x 30	1,344	True
ZeroPad2d	32 x 672 x 33 x 33	0	False
BatchNorm2d	32 x 672 x 15 x 15	1,344	True
Identity	32 x 672 x 1 x 1	0	False
Identity	32 x 28 x 1 x 1	0	False
Identity	32 x 672 x 15 x 15	0	False
BatchNorm2d	32 x 192 x 15 x 15	384	True
MemoryEfficientSwish	32 x 28 x 1 x 1	0	False
Identity	32 x 192 x 15 x 15	0	False
BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
ZeroPad2d	32 x 1152 x 19 x 19	0	False

BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
Identity	32 x 1152 x 1 x 1	0	False
Identity	32 x 48 x 1 x 1	0	False
Identity	32 x 1152 x 15 x 15	0	False
BatchNorm2d	32 x 192 x 15 x 15	384	True
MemoryEfficientSwish	32 x 48 x 1 x 1	0	False
Identity	32 x 192 x 15 x 15	0	False
BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
ZeroPad2d	32 x 1152 x 19 x 19	0	False
BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
Identity	32 x 1152 x 1 x 1	0	False
Identity	32 x 48 x 1 x 1	0	False
Identity	32 x 1152 x 15 x 15	0	False
BatchNorm2d	32 x 192 x 15 x 15	384	True
MemoryEfficientSwish	32 x 48 x 1 x 1	0	False
Identity	32 x 192 x 15 x 15	0	False
BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
ZeroPad2d	32 x 1152 x 19 x 19	0	False
BatchNorm2d	32 x 1152 x 15 x 15	2,304	True
Identity	32 x 1152 x 1 x 1	0	False
Identity	32 x 48 x 1 x 1	0	False
Identity	32 x 1152 x 15 x 15	0	False
BatchNorm2d	32 x 192 x 15 x 15	384	True
MemoryEfficientSwish	32 x 48 x 1 x 1	0	False

```

-----
Identity          32 x 192 x 15 x 15  0          False
-----
BatchNorm2d      32 x 1152 x 15 x 15 2,304      True
-----
ZeroPad2d        32 x 1152 x 17 x 17  0          False
-----
BatchNorm2d      32 x 1152 x 15 x 15 2,304      True
-----
Identity          32 x 1152 x 1 x 1   0          False
-----
Identity          32 x 48 x 1 x 1     0          False
-----
Identity          32 x 1152 x 15 x 15  0          False
-----
BatchNorm2d      32 x 320 x 15 x 15  640        True
-----
MemoryEfficientSwish 32 x 48 x 1 x 1     0          False
-----
Identity          32 x 320 x 15 x 15  0          False
-----
BatchNorm2d      32 x 1280 x 15 x 15 2,560      True
-----
AdaptiveAvgPool2d 32 x 1280 x 1 x 1   0          False
-----
Dropout           32 x 1280           0          False
-----
Linear            32 x 1000           1,281,000  True
-----
MemoryEfficientSwish 32 x 1280 x 15 x 15  0          False
-----

```

```

Total params: 1,323,016
Total trainable params: 1,323,016
Total non-trainable params: 0

```

```

Optimizer used: <function Adam at 0x7f318eebaef0>
Loss function: FlattenedLoss of LabelSmoothingCrossEntropy()

```

```

Callbacks:
- TrainEvalCallback
- Recorder
- ProgressCallback

```

```
[22]: learn.fine_tune(30)
```

```
<IPython.core.display.HTML object>
```

<IPython.core.display.HTML object>

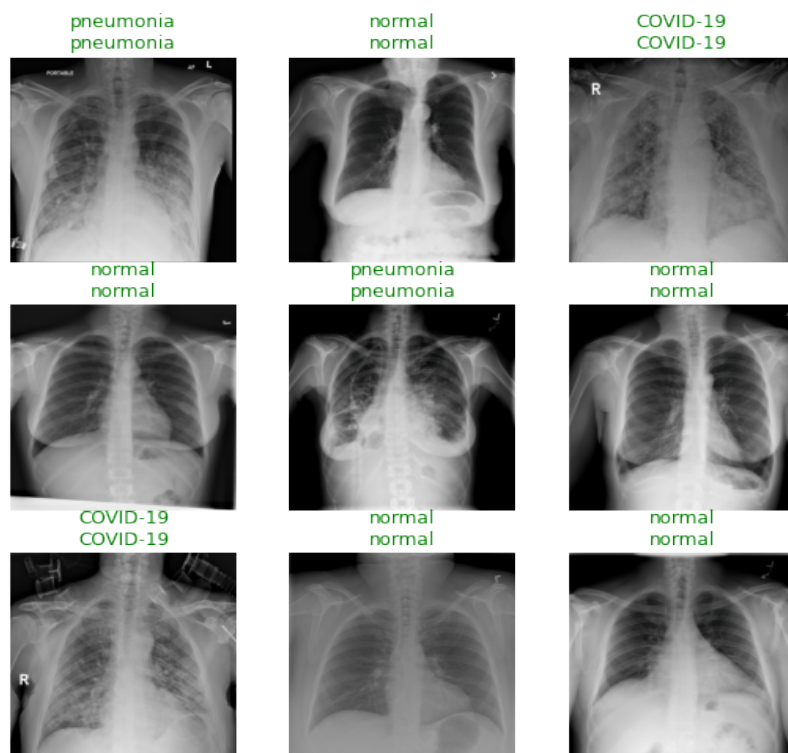
```
[24]: learn.save('covidxraynet')
```

```
[24]: Path('models/covidxraynet.pth')
```

1.5 Interpretation

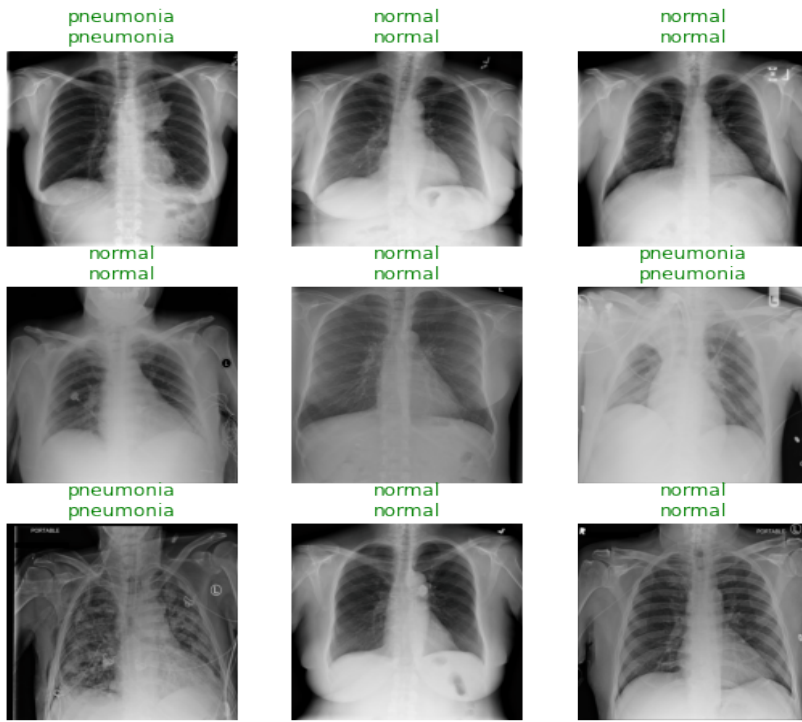
```
[25]: learn.show_results()
```

<IPython.core.display.HTML object>



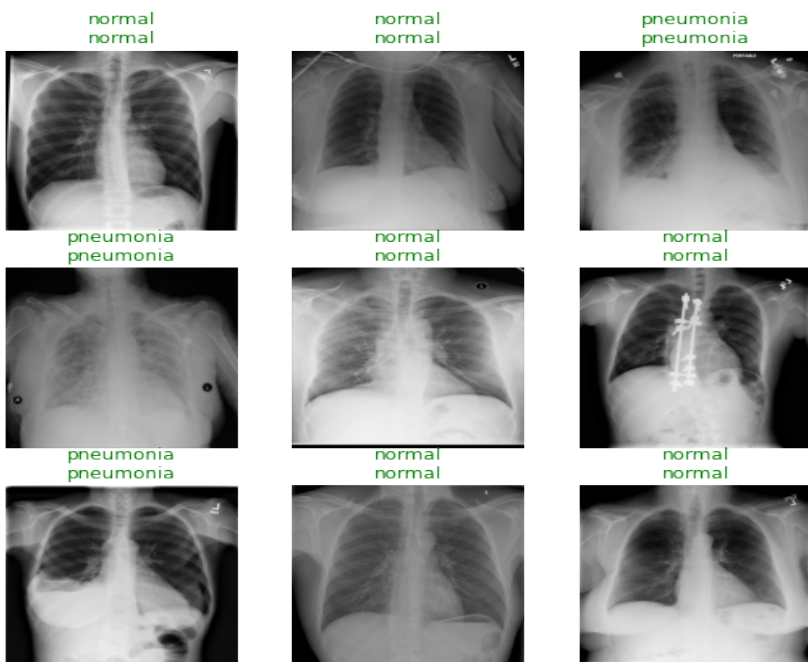
```
[26]: learn.show_results()
```

<IPython.core.display.HTML object>



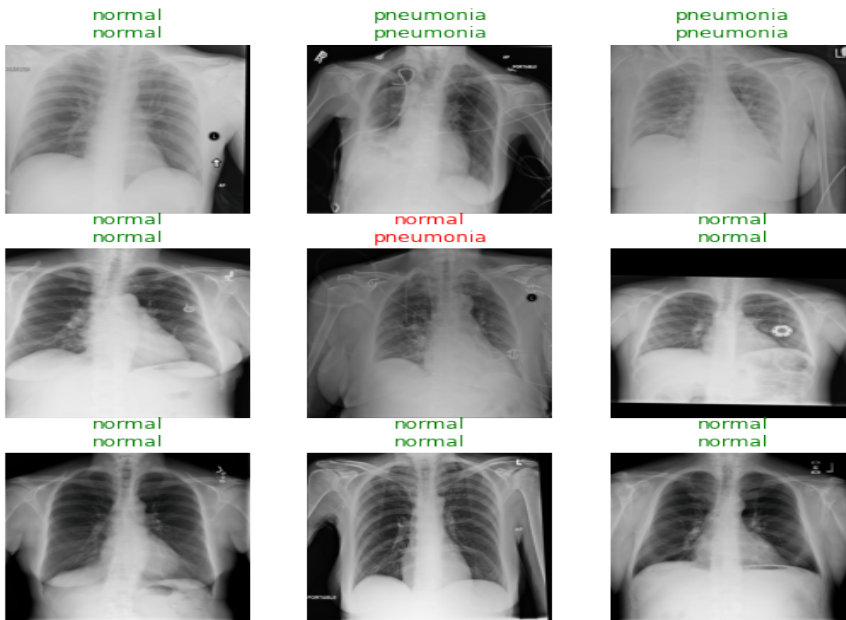
```
[27]: learn.show_results()
```

<IPython.core.display.HTML object>



```
[27]: learn.show_results()
```

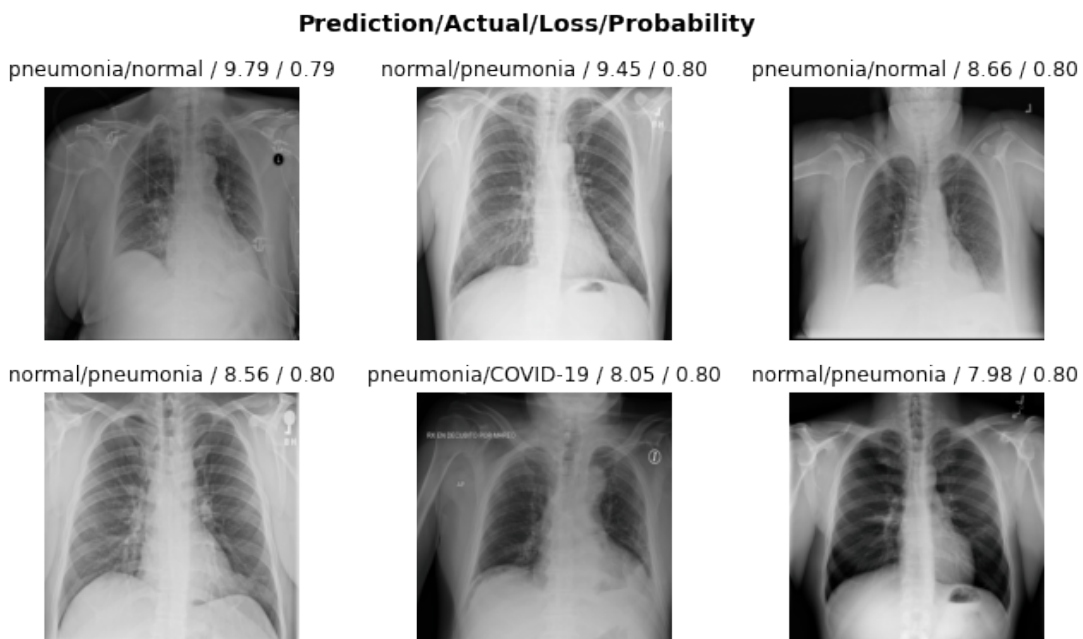
<IPython.core.display.HTML object>



```
[29]: interp = Interpretation.from_learner(learn)
```

<IPython.core.display.HTML object>

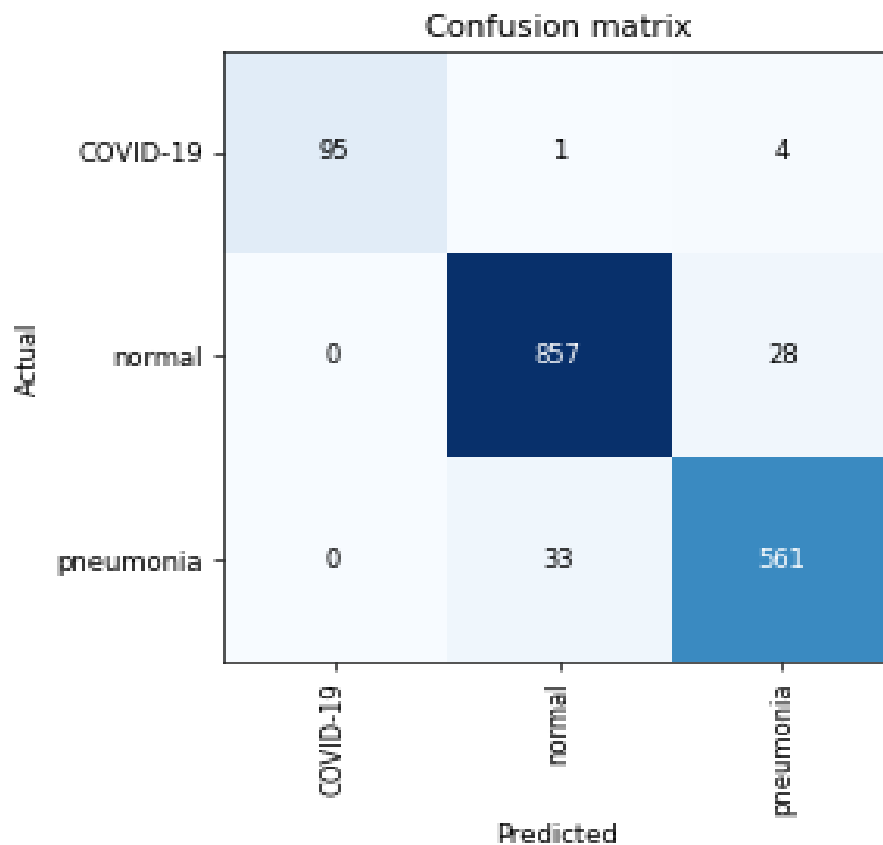
```
[30]: interp.plot_top_losses(6, figsize=(11,6))
```



```
[34]: interp2.most_confused(min_val=1)
```

```
[34]: [('pneumonia', 'normal', 33),  
      ('normal', 'pneumonia', 28),  
      ('COVID-19', 'pneumonia', 4),  
      ('COVID-19', 'normal', 1)]
```

```
[32]: interp2.plot_confusion_matrix(figsize=(5,5), dpi=60)
```



```
[33]: interp2.confusion_matrix()
```

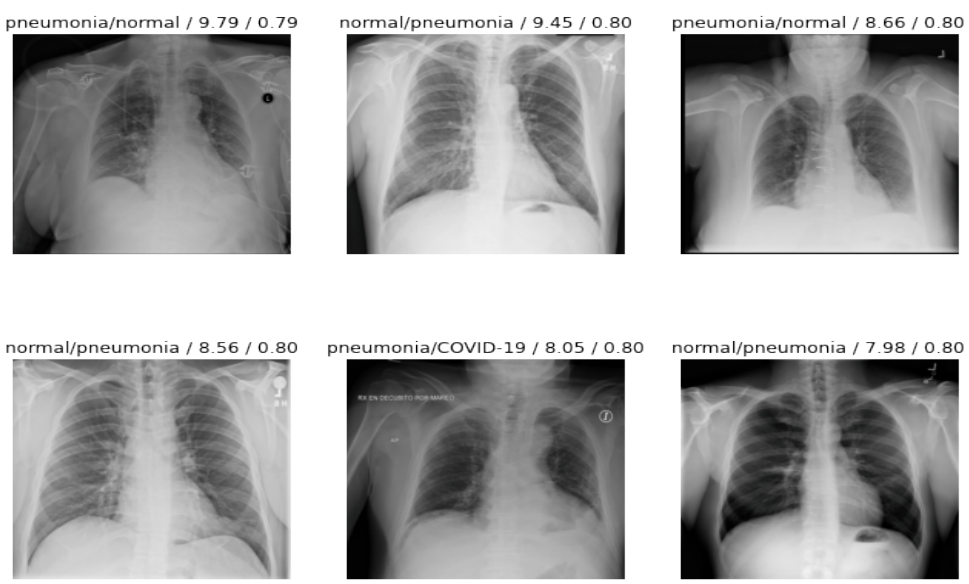
```
[33]: array([[ 95,   1,   4],  
        [  0, 857,  28],  
        [  0,  33, 561]])
```

```
[34]: interp2.most_confused(min_val=1)
```

```
[34]: [('pneumonia', 'normal', 33),  
      ('normal', 'pneumonia', 28),  
      ('COVID-19', 'pneumonia', 4),  
      ('COVID-19', 'normal', 1)]
```

```
[35]: interp.plot_top_losses(6, figsize=(11,9))
```

Prediction/Actual/Loss/Probability



```
[36]: interp.plot_top_losses(9, figsize=(11,9))
```

Prediction/Actual/Loss/Probability

