Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Master Thesis

Deep multiple-instance learning for detecting multiple myeloma in CT scans of large bones

Bc. Vojtěch Mach

Supervised by
Dr. rer. nat. Jan Hering

Submitted in August, 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Mach Vojtěch**	Personal ID number:	**420955**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Branch of study:	**Artificial Intelligence**		

II. Master's thesis details

Master's thesis title in English:

Deep multiple-instance learning for detecting multiple myeloma in CT scans of large bones

Master's thesis title in Czech:

Hluboké učení z více instancí pro detekci mnohočetného myelomu v CT snímcích dlouhých kostí

Guidelines:

Multiple-instance learning (MIL) is a weakly supervised machine learning approach to deal with sparsely annotated data. Such data occur often in the context of medical data analysis, where the diagnosis (global label) is given naturally during the clinical routine but voxel-level annotations are not provided. The MIL training paradigm is typically included into a CNN-model via an aggregation layer prior to loss function computation.
The primary goal of this thesis is to evaluate the feasibility of deep learning MIL for automated classification of CT scans of femur bones. The objectives of the thesis are
1) Prepare the image dataset for region-based training [3, 5]
2) Study state-of-the-art (CNN-based) MIL approaches [1, 4]
3) Implement and evaluate selected CNN-MIL methods on augmented datasets (to have voxel-level labels)
4) Adapt/extend the selected CNN-MIL approach to the thesis' domain
5) Evaluate the best CNN-MIL model on real clinical dataset and compare against the latest results by traditional MIL classifier [2]

Bibliography / sources:

[1] Durand, T. et al., WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks, 2016 IEEE CVPR, pp. 4743–4752.
[2] Hering, J et al., Detecting Multiple Myeloma via Generalized Multiple-Instance Learning, SPIE Medical Imaging 2018.
[3] Klein, A. et al., Automatic bone segmentation in whole-body CT images.2019 JCARS 14, 21–29.
[4] Kraus et al., Classifying and segmenting microscopy images with deep multiple instance learning. 2016, Bioinformatics 32, i52–i59.
[5] Martínez-Martínez, F. et al., Fully Automated Classification of Bone Marrow Infiltration in Low-Dose CT of Patients with Multiple Myeloma Based on Probabilistic Density Model and Supervised Learning. 2016, Comput. Biol. Med. 71, 57–66.

Name and workplace of master’s thesis supervisor:

Jan Hering, Dr. rer. nat., Biomedical imaging algorithms, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **04.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

Jan Hering, Dr. rer. nat.		prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Abstract

The employment of computer aided diagnosis (CAD) systems for interpretation of medical images has become an increasingly popular topic with the arrival of modern machine learning algorithms. Convolutional neural networks perform exceptionally well nowadays in various pattern recognition tasks including image classification. In this thesis we examine the capabilities of a convolutional neural network binary classifier as a CAD system for detection of abnormalities in CT images of femurs. We focus on the diagnosis of multiple myeloma characterized by symptomatic bone marrow lesions commonly observable through computer tomography screening. Different approaches to the problem including multiple instance learning (MIL) were tested. The classifier showed a solid performance in our fully supervised experimental setting, it however exhibits a serious inability to learn from multiple instances. We conclude that the proposed neural model needs a stronger error signal in order to converge in the standard MIL setting and suggest potential improvements for further work in this area.

Keywords: computer aided diagnosis, multiple myeloma, machine learning, deep learning, convolutional neural networks, multiple instance learning

Abstrakt

S nástupem moderních algoritmů strojového učení vzrostla popularita tématu automatické interpretace výstupů zobrazovacích metod v medicíně pomocí počítačů. Konvoluční neuronové sítě v současné době excelují v mnoha oblastech strojového vidění včetně rozpoznávání obrazu. V této diplomové práci zkoumáme možnosti využití konvolučních sítí jako diagnostického nástroje pro detekci abnormalit v CT snímcích stehenních kostí. Zaměřujeme se na diagnózu mnohočetného myelomu pro nějž jsou charakteristické viditelné léze v kostní dřeni, které lze pozorovat při vyšetření pomocí počítačové tomografie. Bylo otestováno několik různých přístupů včetně učení z více instancí. Náš klasifikátor podává spolehlivý výkon v experimentech s plně supervizovaným učením, vykazuje ovšem zásadní neschopnost konvergence při učení z více instancí. Předpokládáme, že náš navrhovaný neuronový model potřebuje ke konvergenci silnější chybovou odezvu a na toto téma navrhujeme budoucí možná vylepšení.

Klíčová slova: analýza medicínských snímků, mnohočetný myelom, strojové učení, hluboké učení, konvoluční neuronové sítě, učení z více instancí

Declaration:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, Friday 14th August, 2020

Vojtěch Mach

Acknowledgements

I sincerely thank my supervisor Jan Hering for his guidance during the whole process of writing this thesis. I am grateful for the opportunity to work on this interesting topic and I highly appreciate his valuable advice which helped me gain a deeper understanding of the researched areas. I also want to express my deepest gratitude to my family and close ones for their relentless support throughout my studies. This would never have been possible without your help.

Contents

Introduction 1

1 Background **3**

1.1 Medical background 3

1.1.1 Medical screening 3

1.1.2 X-ray computer tomography 4

1.1.3 Multiple myeloma 7

1.2 Technical background 9

1.2.1 Machine learning 9

1.2.2 Neural networks 13

1.2.3 Convolutional neural networks 19

1.2.4 Deep learning 27

1.2.5 Model evaluation 32

1.2.6 Multiple instance learning 35

2 Related work **39**

2.1 Task-related 39

2.2 Model-related 42

3 Dataset **45**

3.1 Structure 45

3.1.1 Original dataset 45

3.1.2 Derived dataset 46

3.2 Data cleaning 47

3.2.1 Femur skew correction 47

3.2.2 Bone implants 48

3.2.3 Removing joints 48

3.2.4 Removing air voxels 51

3.3 Data flow 52

3.3.1 Creating femur patches 53

3.3.2 Augmentations 54

3.3.3 Transformations 57

4 Implementation **59**

4.1 Software design 59

4.1.1 Tools 59

4.1.2 Data-manipulation hierachy 60

4.2 Classifier design 60

4.2.1 Base network 61
4.2.2 MIL modifications 61
4.3 Training settings 64

5 Experiments **67**
5.1 Input arguments 67
5.2 General settings 68
5.3 Supervised learning experiment 69
5.3.1 Training 69
5.3.2 Evaluation 70
5.4 Multiple instance learning experiment 70
5.4.1 Standard MIL 70
5.4.2 Transfer learning 72

Conclusion **75**

Bibliography **77**

Appendix A **82**

Appendix B **85**

Introduction

Artificial intelligence and machine learning have attracted an unprecedented attention in recent years. Even though the origins of AI research date back to 1950s, it was not until this century, where enormous data collections and high-performance computational hardware became widely available and gave rise to many successful real-world applications. Various intelligent systems begin to infiltrate everyday lives of our society on multiple levels ranging from common consumer electronics to weather forecasts, autonomous vehicles or surveillance systems. Given the current pace of innovation in this area, it is not surprising that general public believes that artificial intelligence will be the driving force of next industrial revolution.

In the past, computer scientists have studied various approaches to AI based on different mathematical principles although in the modern era of data-driven businesses and powerful hardware, many researchers have shifted their primary focus to statistical methods of inference which seem to perform better than others in the current environment.Nowadays statistical machine learning benefits from growing computational power and vast amounts of data stored and shared over the internet every day.

Technological and scientific advances of the last decade has brought a great popularity to deep learning – an area of statistical machine learning focused on optimising very complex neural networks using huge quantities of data for the purpose of solving specific abstract tasks from various fields. It has been particularly successful or even revolutionary in computer vision and natural language processing. Deep learning of convolutional neural networks, which deals with models specifically modified to pick up spatial and temporal relations between patterns in inputs, has dominated image recognition competitions and is currently regarded as the state-of-the-art approach for variety of computer vision tasks. Despite many impressive results, it has also been a subject of criticism mainly addressing its undesirable reliance on large datasets and poor interpretability of result models.

In recent years, deep learning has been applied in a wide range of professions including medicine. One such example of such interdisciplinary fusion is computer-aided diagnosis (CAD) – a computer system used by doctors and medical experts such as radiologists as a supportive diagnostic tool for interpretation of screening examination results. These systems perform image recognition in order to localize and describe possible areas of interest in the image and therefore provide the professional with an additional diagnostic input.

The goal of this thesis is to use the aforementioned deep learning methods to design and implement a prototype of CAD system providing a diagnosis of

multiple myeloma based on visible symptomatic bone marrow lesions in CT images of femurs. We approach this by first researching suitable methods focused on related work in the area followed by implementation of our proposed solution. For that purpose we design a binary convolutional neural network classifier fit for our data and perform extensive data cleaning procedures which aim to help alleviate the model optimisation process. In the second part of the thesis, we propose two consecutive experiments with different settings – a traditional fully supervised learning experiment followed by a multiple instance learning (MIL) experiment. Both approaches are evaluated and the latter is compared to the results of related MIL models dealing with similar tasks. Finally, a conclusion regarding the method and its results is made and possible means of improvement are suggested.

1 | Background

Contents

1.1 Medical background 3
1.1.1 Medical screening 3
1.1.2 X-ray computer tomography 4
1.1.3 Multiple myeloma 7
1.2 Technical background 9
1.2.1 Machine learning 9
1.2.2 Neural networks 13
1.2.3 Convolutional neural networks 19
1.2.4 Deep learning 27
1.2.5 Model evaluation 32
1.2.6 Multiple instance learning 35

The goal of this thesis is to propose and prototype a computer program which performs an automatic diagnostic of multiple myeloma from CT images using selected machine learning methods. This chapter is dedicated to introduction to both medical and technical notions and description of terminology used throughout the text of the thesis.

1.1 Medical background

This section provides a basic description of medical foundations of our project and explains the terminology used in the text. It also gives a brief overview of the history and present of computation in medical imaging with an accent on computer tomography. Lastly it focuses on multiple myeloma, a cancerous plasma cell disorder resulting in lesions and infiltration of bone marrow which is the focus of the project.

1.1.1 Medical screening

Medical imaging has gone through a remarkable development in the 20th century. The foundations of radiological imaging were laid by Wilhelm Röntgen in the end of the 19th century when he discovered X-rays during his research of radiation. In 1895 he captured the famous picture of the skeleton of his wife's hand producing the first recorded radiogram in history. Since then, medical imaging has reported a massive progress most notably with the arrival of computational power in the digital era as illustrated on figure 1.1.

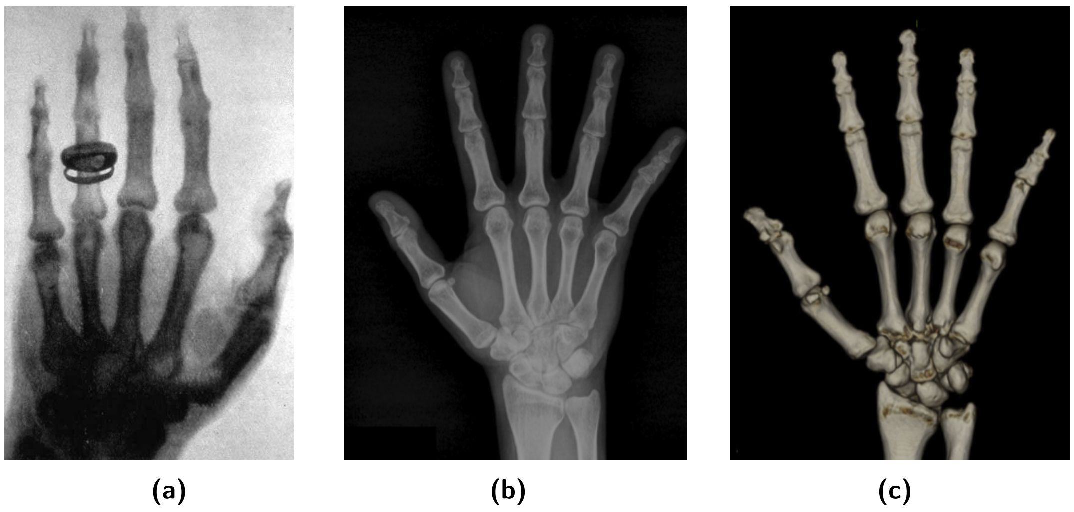


Figure 1.1: A comparison of the Röntgen's 1895 radiogram (a) and its modern counterpart obtained from a CT scan (b) including its 3D computer reconstruction (c).

1.1.2 X-ray computer tomography

History

The first commercially used CT scanner was developed by Sir Godfrey Hounsfield in the early 1970s while conducting research for British company EMI Ltd. According to a popular belief, British music band Beatles indirectly helped finance Hounsfield's research, since at the time EMI was also the band's record producer. However, the extent of their actual contribution was doubted [1] and so the story remains unverified. In 1979 Hounsfield was awarded the Nobel prize in physiology and medicine for his invention.

Mechanism

A CT scanner acquires a cross-sectional 2D image by repeatedly casting X-ray beams from an emitter to a photosensitive detector while rotating around the scanned object. By moving the object through the scanner, different cross-sections are acquired and a stack of these slices creates the final 3D CT image.

Computer tomography has evolved significantly since the first scan performed in 1973. Output image resolution as well as the scanning speed were both improved by several orders of magnitude. For illustration, a slice resolution went from the original 80×80 up to 512×512 in common clinical screenings or even to 1024×1024 in current state-of-the-art scanners [2]. New generations of CT scanners are also able to acquire a high resolution whole-body scan in a much shorter amount of time then before. The duration of a single scanning was reduced from > 5 minutes to a couple of seconds [3].

Hounsfield scale

Hounsfield units (HU), also termed CT numbers, are quantitative measurement units used to interpret the radiodensity of bodily tissues and other materials in the scanned subject. Hounsfield scale is a linear transformation of the attenuation measurements into a scale where radiodensity of water and air of standard pressure is defined as 0 HU and -1000 HU respectively. The transformation is defined as follows:

$$HU = 1000 \times \frac{\mu_{\mathrm{x}} - \mu_{\mathrm{water}}}{\mu_{\mathrm{water}} - \mu_{\mathrm{air}}}, \tag{1.1}$$

where μ_x denotes an attenuation coefficient of a material or tissue x.
In the context of radiography, the attenuation coefficient associated with a material characterizes the ability of an X-ray beam to penetrate the material, where larger coefficient means larger portion of the beam which passes through without being absorbed or reflected. This implies that the attenuation value is directly dependent on the the applied radiation energy which would complicate the interpretation of the results. This is conveniently solved by the Hounsfield scale transformation which removes this dependence and as a result, HU values of different tissues are constant and invariant to the input voltage. Common materials and body tissues with their respective Hounsfield units are recorded in table 1.1.

Tissue or material	Common values (HU)
Air	-1000
Fat	-130 to -100
Water	0
Muscle	$+10$ to $+40$
Brain matter	$+20$ to $+45$
Bone marrow	-150 to -50
Cancellous bone	$+300$ to $+700$
Cortical bone	$+1800$ to $+1900$
Metal implants (alloys)	$> +2000$
Raw metals (steel, gold, silver)	$\sim 10^5$

Table 1.1: Table of commonly reported CT number measurements of various substances. Presented values were compiled from multiple sources.

The Hounsfield scale is theoretically open-ended, although selected bit-depth of the result gray-scale image defines the range of available values. In practice, commonly used 12-bit color depth encoding represents exactly 4096 values, which satisfies the range of -1000 to $+3071$ Hounsfield units covering all human body tissues including from soft tissues to bones. In examinations of high-density materials like metal implants, an extended 16-bit encoding is often used providing 65536 values which covers substances from air to precious metals [4].

CT image interpretation

According to a recent study [5], human eye is believed to detect around 10 million different colors but only about 30 shades of gray. This estimation shows a great disbalance in human perception of color versus brightness, which is caused by biological structure of human visual system.

As implied by table 1.1, a common clinical CT scan of a human body may contain around 2500 HU values ranging from -1000 HU to 1500 HU, all representing different gradations of gray. Furthermore, conventional 8-bit or 10-bit monitors used in medical environment are only capable of displaying 256 or 1024 different shades of brightness respectively [6]. To distinguish between organs of similar attenuation, the raw values from a CT scan must be mapped to an appropriate range, where the dynamic contrast between such tissues is enhanced – technique commonly referred to as windowing. The window is a chosen interval of CT values and it is adjustable by two parameters – width and level. Window width defines width of the interval of CT values the examiner wishes to display while window level, also refereed to as window center, denotes the midpoint value of the chosen interval. A proper windowing is the key to successful diagnosis and radiologists are trained to apply suitable windowing in different clinical cases. Table 1.2 presents the most common window widths and levels used in imaging of different body parts.

Body part	Window level (L) and width (W) (values in HU)	
Brain	L: 30	W: 100
Heart	L: 35	W: 30
Liver	L: 65	W: 10
Lung	L: -700	W: 400
Fat	L: -70	W: 250
Cancellous bone	L: 120	W: 200
Cortical bone	L: 1000	W: 1500

Table 1.2: A common window levels and widths used in CT examination of different body parts and organs. Source: [7]

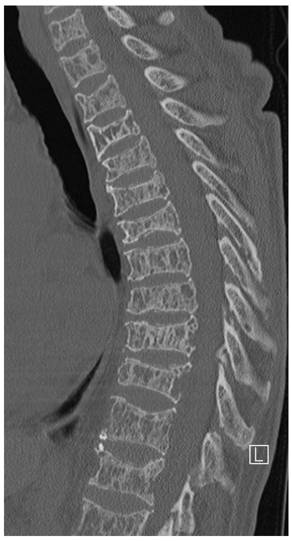
An increased awareness of harmful effects of repeated radiation exposure has triggered the development of alternative screening methods. Although, low-dose CT scans are currently performed [3], in some cases such as during pregnancy it is unwanted to expose patients to any level radiation. Therefore, methods based on different physical principles started to emerge. Most notable innovations of the 20th century in this area include diagnostic sonography and magnetic resonance imaging (MRI) described in the following paragraphs.

Magnetic resonance provides a diagnostic power comparable to a CT scan without emitting X-rays. MRI scanners use electromagnetism and radio waves to create strong magnetic field and excite certain atoms in the imaged object. The resultant spin polarization of the atoms induces an electromagnetic radiation that is detected by the coils in the machine and transformed into the final image. In clinical screening, hydrogen is often targeted since it is omnipresent in the majority of living organisms mostly in watery and fat tissues. For this reason MRI scans, unlike CT or X-ray scans, are especially sensitive to soft tissues. Exposing the human body to a magnetic field is considered harmless, however there are limitations to this technique in terms of cost, comfort and materials which may be a subject to the examination. Compared to CT scans, it is still a relatively expensive diagnostic method, the scanning procedure takes quite a long time and it might be uncomfortable especially for claustrophobic patients due to the loud present noise and constraint movement inside the narrow tube where the patient is laid.

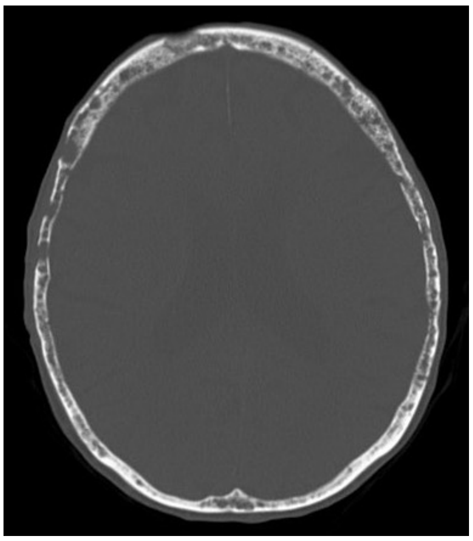
1.1.3 Multiple myeloma

Multiple myeloma (MM) is a malignant cancerous plasma cell disorder which causes proliferation of clonal cells in bone marrow. It is the second most common hematologic malignancy and the most frequent bone cancer [8]. The disease is most often present in elderly, recorded median ages of diagnosed patients in 2016 settle around 70 years [9, 10]. Studies suggest an incidence of six to seven positive cases in 100000 per year in USA and Europe, with men being more likely to be diagnosed than women [9]. The cause of multiple myeloma is not fully understood, probable factors favoring the occurrence of the disease include radiation, pesticides, chronic infections and obesity [10]. Common indications of MM include bone pain, fatigue or anemia, although in many patients symptoms are latent or vague until more advanced stages of the disease [11].

Diagnosis of multiple myeloma is currently based on blood or urine tests. In advanced stages of the disease lytic lesions located in bone marrow are observed under medical screening. Therefore, CT or MRI scans of patients examined for other reasons might be evaluated by a reliable CAD system for presence of different injuries or diseases including multiple myeloma.



(a) Sagittal image of lumbar vertebrae infiltrated by multiple myloma. Source: [12].



(b) An axial CT image of so called *raindrop skull* representing a severe case of MM with numerous lytic lesions. Source: [13].

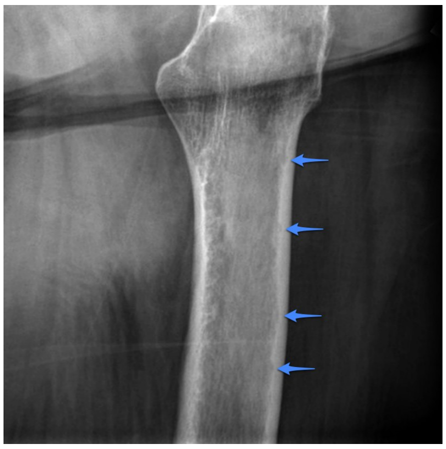
Figure 1.2: CT images showing infiltration in vertebrae (a) and in a skull (b).

Symptomatology

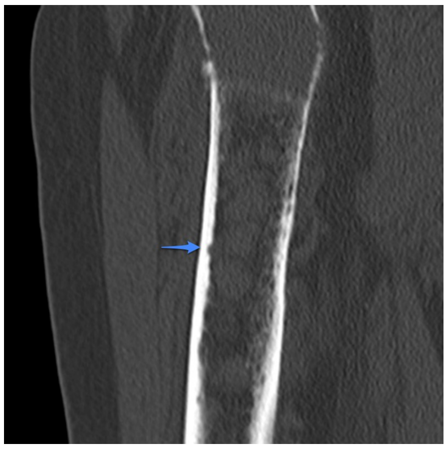
Bone pain and fractures are common symptoms caused by destroyed bone structure by adjacent cancerous cells. In the majority of cases the first infiltration appears in axial skeleton, therefore vertebrae, ribs, skull or pelvis [14] are usually among the first areas developing visible symptomatic manifestations. The infiltration often results in bone pain which is reported by the majority of MM patients [15]. These deformations can be observed through medical imaging in a form of osteolytic lesions spread over bone marrow [15]. The lesions are characteristic areas of increased osteolysis with highly suppressed or absent osteoblast function [8]. A generally recommended diagnostic imaging method is a low-dose whole-body CT scan, alternatively other more sensitive methods such as MRI or PET might be used to reveal other MM lesions in soft tissues [10, 16]. Under CT screening, the bone marrow infiltration usually appears in a form of small localized islets of changed density. According to a recent study, the radiodensity of the medullary lesions was found to range from −90 HU to 70 HU [17].

The osteolytic lesions are particularly relevant for this thesis since they represent the basis of our practical project. Their role in the system prototype implementation phase is further explained in chapter 3. Lesions present in severe multiple myeloma cases are clearly distinguishable as shown on figure 1.2.

In serious stages of the disease, one might also observe so called endosteal scalloping, an etching in the inner wall of cortical bone caused by a growth of neighboring lesions (Fig. 1.3). Scallops have a destructive effect on bone rigidity and may lead to fractures.



(a) Multiple scallops in cortical femur.



(b) Endosteal scallops and pronounced lytic lesions in a severe case of MM.

Figure 1.3: X-ray (a) and CT (b) screenings of endosteal scalloping in femurs caused by multiple myeloma. Affected areas are indicated with arrows. Source: [12].

1.2 Technical background

Following sections provide an insight to techniques used in the development of the thesis project. General notions and terminology used in the field of machine learning are explained in the beginning. The main attention is paid to neural networks, especially to convolutional networks and deep learning. The final section explains the notion of multiple instance learning, its origins and meaning in the context of our project.

1.2.1 Machine learning

Machine learning (ML) is a field of artificial intelligence that aims to extract knowledge, infer facts and find relationships in collections of data with an intent to generalize on new unseen data of the same domain, predict development of a modelled process or provide other reasoning. For that it exploits a number of mathematical concepts like statistics, logic, linear algebra or numerical optimisation and offers algorithms which are designed to run effectively on a computer. The mechanisms of these algorithms resemble human learning process, hence the name.

History

The origins of machine learning date back to 1950s but its true potential became apparent in the current century, especially in the second decade. A considerable portion of ML practice overlaps with data mining – a formerly used technique attempting to extract knowledge from large datasets. Even though both share a few common techniques to pursue similar goals, nowadays data mining may already be considered a subset of machine learning since the scope of machine learning grew immensely over recent years. The main difference lies in used algorithms – while data mining works mostly with clustering algorithms in order to categorize data, ML deals with much broader set of tasks ranging from self-improving algorithms to image recognition systems or even generative models which learn to sample an entirely new data from a given domain.

Paradigms

Structuring the whole field of machine learning into disjunctive sections is not trivial because many interdisciplinary ML approaches already exist, but the following list provides a basic categorization of modern machine learning algorithms into three high-level paradigms distinguished by the nature of the task, structure of available data and recommended optimisation processes.

1. **Supervised learning**

   The learner is provided annotated data usually in a form of data-label pairs. Depending on the annotation coverage we distinguish between different types of supervision. For instance, weakly supervised learning studies all kinds of tasks with incomplete dataset annotations. On the contrary, fully supervised learning is a simpler setting where every data point is assigned a label. The systems learning under this paradigm repeatedly cycle through the data to learn probabilistic mapping from the input space to the space of outputs. Output predictions are then compared to the sample outputs and an error is measured which is subsequently used to adjust the system properties in order to yield better predictions in future iterations. Supervised learning is widely used for plethora of classification and regression tasks and it recently achieved a notable industrial success in combination with neural networks, which are thoroughly described in 1.2.2.

2. **Unsupervised learning**

   There are many real-world tasks based on data collections which lack annotations for various reasons – they might be anonymized for instance or not exist at all. In such cases, clustering algorithms represent a popular approach to provide reasoning and inference over such data. These techniques are widely employed in market applications such as product recommendation systems.

3. **Reinforcement learning**

   Reinforcement learning studies the behaviour of an abstract agent and its interactions with a surrounding environment. The agent learns to optimize its behaviour continuously from feedback signals received upon performing actions which affect the environment as depicted on figure 1.4. Problems in this area are usually modelled as Markov decision processes and applications include robot control, expert-level game systems etc.

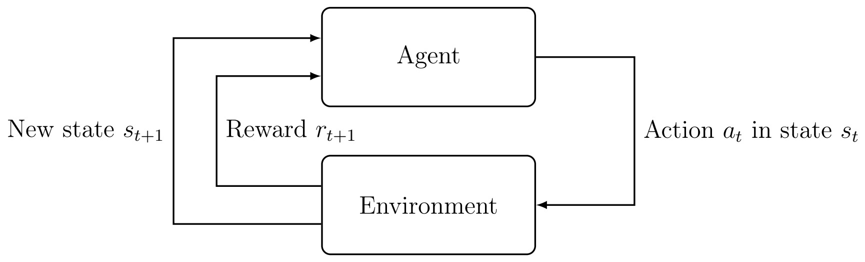


Figure 1.4: A schematic image of the action-reward mechanism in a typical reinforcement learning setup.

These three categories roughly encompass most of the currently popular machine learning techniques though, it may be rather oversimplified as new approaches overlapping and fusing multiple paradigms emerge nowadays, but it will suffice for the scope of this thesis.

Statistical machine learning

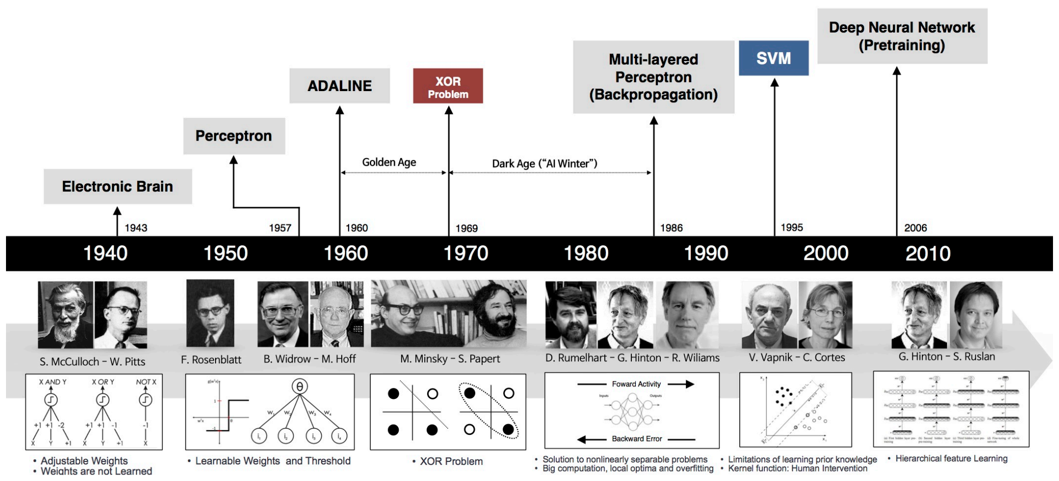
A comparably important distinction can be made regarding the different mathematical fundamentals used in the field. This addresses not solely machine learning but the whole AI which was approached from different technical and philosophical angles in the past. Two main competing ideological trends were symbolic AI relying on mathematical logic and reasoning over explicitly stated rules and non-symbolic AI focusing mostly on statistical inference over data.

While the symbolic AI was in the center of focus for many years in the previous century, probabilistic methods became increasingly accurate and successful in the modern world where shortage of neither data nor computational resources is a limiting factor. The current century has seen a tremendous increase in amounts of all kinds of data stored, shared and transferred over the internet. The rise of personal computers followed by the era of smartphones and other multimedia gadgets, worldwide internet access, social websites, internet of things, cloud storage etc. all contributed to this phenomenon which resulted in the current era of data-driven businesses.

Additional benefits of statistical models include their robustness against unknown or malformed inputs and the ability to increase their accuracy. Algorithms which learn rules and concepts automatically from observations may be improved by simply supplying more data, whereas rule-based systems can only be improved by providing more complex rules, which is a substantially more difficult procedure.

Current state of AI

Artificial intelligence in general has gone through a few periods of optimistic and pessimistic eras in its long history of more than seventy years as shown on figure 1.5. Unfulfilled expectations or funding cuts have been some of the triggers for periods of lost interest and decreased enthusiasm about its abilities aptly coined as *AI winter*.



Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
ADALINE
XOR Problem
Perceptron
Golden Age
Dark Age ("AI Winter")
Electronic Brain
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch – W. Pitts
F. Rosenblatt
B. Widrow – M. Hoff
M. Minsky – S. Papert
D. Rumelhart – G. Hinton – R. Wiliams
V. Vapnik – C. Cortes
G. Hinton – S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Figure 1.5: Illustration of the timeline and milestones of the AI research. Source: [18]

Its renaissance which we observe today is caused mainly by the enormous quantities of available data and increase in computational power which are two essential ingredients for fruitful applied statistical machine learning. Additionally, the field has been revolutionized by the arrival of rediscovered methods like deep learning and complex neural networks which flourished in the modern environment. Over the course of the past decade they have become extremely popular

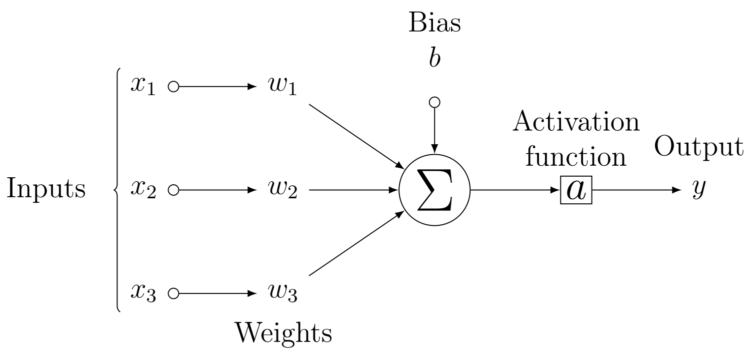


Figure 1.6: A schematic picture of a single neuron unit.

after a breakthrough success they have achieved in various tasks across different areas of AI. These methods as well as the events leading to their current position are explained thoroughly in the following sections.

1.2.2 Neural networks

An artificial neural network (ANN) is a parametrized connectionist model of a function approximator f that performs a mapping $f(\boldsymbol{x}, \boldsymbol{\theta}) : \boldsymbol{x} \to y$, where $x \in X$ is an element from an input space X, $y \in Y$ is an element of an output space Y and $\boldsymbol{\theta}$ is the set of parameters. The system was inspired by the structure of a biological brain where interconnected neurons receive and output signals using electrical impulses. It can be interpreted in a form of a directed graph structured in a sequence of layers performing linear and non-linear transformations to given inputs. This analogy is convenient for schematic illustrations of different architectures.

Neuron structure

A building block of every neural network is called neuron and it is a simple function that calculates dot product of an input vector $\mathbf{x}$ with a set of weights $\mathbf{w}$ and a bias b according to the formula

$$y = a(\mathbf{w}^T\mathbf{x} + b),$$

where a is a non-linear transformation function also referred to as activation. The bias b is usually incorporated in the weights vector for faster computations and simpler notation. A graphical representation of neuron is shown on figure 1.6.

Neurons in a neural network are arranged in multiple layers where outputs of one layer are the inputs of the next layer with the exception of the first and the last layer. In a fully connected neural network, every neuron from layer l takes output of every neuron from layer $l - 1$ as its input.

For historical reasons a single neuron with a step function as activation function is also called *perceptron* or single-layer perceptron. Similarly, since neural networks are structured layers of perceptrons they are sometimes called multi-layer perceptrons (MLP). This connectionist framework is quite flexible for representation of other machine learning models as well. Logistic regression, a popular mathematical model of a binary classifier, can be also expressed and optimized in this context as a single neuron with sigmoid activation function. The same model with identity activation then represents a linear regression.

Typology

A design and architecture of ANNs is an extensive area of study with multiple branches which is beyond the scope of this thesis, but we may present the two most prominent types of networks and highlight the contrast between them.

The first type, a feedforward neural network, is the simplest network architecture template. These networks can be formalized as directed acyclic graphs (DAG) and they feature multiple consecutive layers of neurons, specifically an input layer followed by one or more hidden layers and an output layer. The main characteristic feature is that the data always flow in a fixed direction from the start in a form of input to the end of the network where outputs are returned (Fig. 1.7).

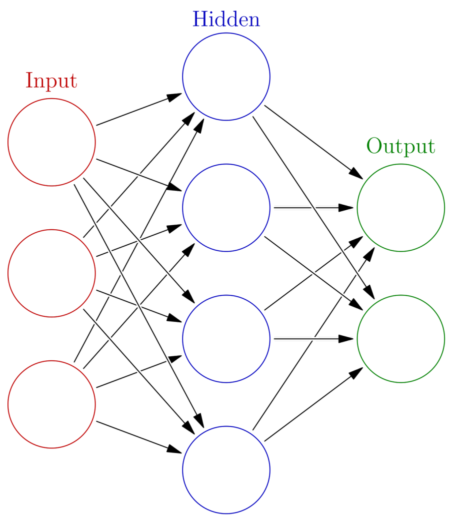


Figure 1.7: A fully connected feedforward neural network with one hidden layer performing a function $f : \mathbb{R}^3 \to \mathbb{R}^2;\ \mathbf{y} = f(\mathbf{x})$. The edges oriented in the direction of output indicate the flow of the tensors through the network.

The number and shape of hidden layers vary depending on the network's purpose. For instance, undercomplete autoencoders are feedforward networks commonly used for input dimensionality reduction and vector embedding and therefore often contain narrower hidden layers resembling an hourglass shape (Fig. 1.8). Complementary, hidden layers in image recognition networks typically grow wider towards the middle of the network followed by narrower layers.

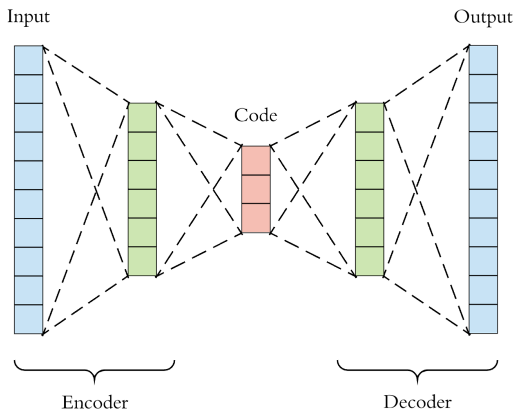


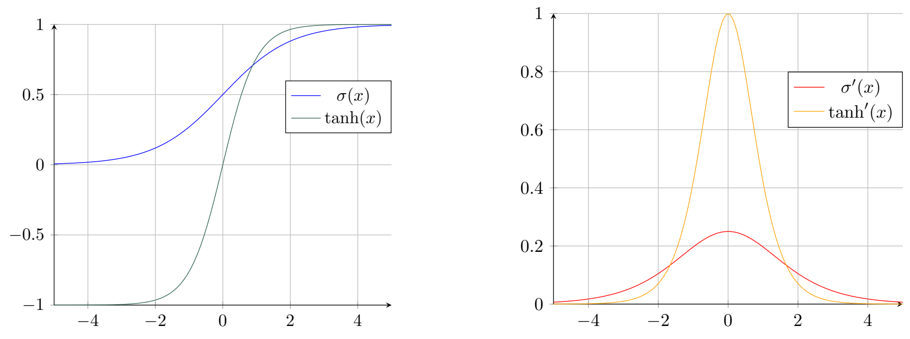
Figure 1.8: Schematic image of a basic undercomplete autoencoder. The model learns to reconstruct an input after passing it through the bottleneck architecture. The middle red-colored layer contains a latent representation of input in a lower-dimensional space. Source: [19].

Despite its simplicity, feedforward architecture is widely popular and well-performing over a variety of domains. It is the basis of the current state-of-the-art models performing image and video recognition and other classification tasks.

The second achitectural template is a recurrent neural network (RNN), which is characterized by a cycle in its graph representation and the data flow is therefore not unidirectional, unlike in the feedforward design. The building block of an RNN is a complex computational neuron-like cell unit which links its output back to itself over the cyclic connection. Due to their recurrent nature, RNNs are usually visualized by unrolling the structure for several timesteps. These networks process sequences of inputs while feeding the current output of the cell back to itself as an additional internal state input along with the next external input in the sequence. The role of the state input is to capture long-distance relationships between patterns in the input sequence and it is often compared to a kind of simplistic memory unit which captures the current state of the network in the context of the modelled system. This property is exceptionally important when dealing with task featuring temporal dimension and logically ordered sequences of inputs, in particular text or video. One of the fields most revolutionized by application of RNNs is natural language processing (NLP) – an area of machine learning which focuses on machine translation, speech recogniton, text understanding and generation.

Activation functions

Activation functions are the essential part of every neural network as they allow the model to fit very complex functions and have a large impact on the optimisation process. A neural network without non-linear transformations is only capable of approximating affine functions since composition of multiple affine transformations is again an affine transformation (proof omitted). This implies that a multi-layer network featuring only linear layers has an identical expressiveness as a single-layer linear perceptron. Such a network, if treated as a classifier, produces a linear decision boundary in the feature space, which means it is only able to perfectly classify a linearly separable data. The `XOR` function is one such example of a simple but linearly non-separable problem. It was this issue which in 1969 triggered an AI winter which lasted for almost two decades [20].



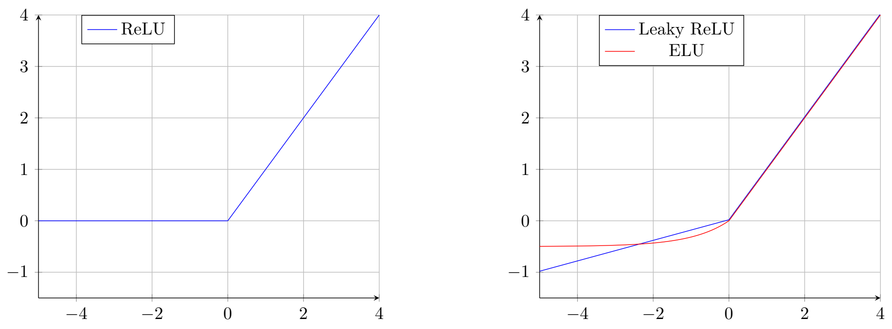
(a) Sigmoid (σ) and hyperbolic tangent (tanh) are both bound and saturating.

(b) Primes of sigmoid and hyperbolic tangent. Maximum gradient value is much higher for the tanh function.

Figure 1.9: Sigmoid and hyperbolic tangent functions (a) and their respective primes (b). Hyperbolic tangent is a superior option for activation function due to its higher maximum gradient value and centering in zero point.

In the past the most used activation functions were sigmoid and hyperbolic tangent (Fig. 1.9). Both of these functions are saturating, which means that they are bound and on their extremities their gradients converge to zero. This complicates the optimisation of neural networks, since small gradient values result in small weight adjustments which consequently inhibits or slows down the process substantially. The sigmoid function was formerly popular for its connection to activation of real biological neurons, however it has a few extra drawbacks apart from the mentioned saturation property. It also has an undesirable property of not being zero-centered. This has reportedly a negative impact on the training process, therefore the hyperbolic tangent was usually recommended instead. Although this issue is mitigated by modern batch optimization methods, it has already been widely replaced by ReLU activation and its modifications.

ReLU is a much simpler piecewise linear function defined as $max(x, 0)$, which is much computationally efficient than the previously mentioned functions. It does not suffer from the saturation issues and its gradient is 1 for all positive inputs, which is also beneficial. Nevertheless, similar problems might arise due to its flat part where gradient is zero which might theoretically cause some neurons to degenerate and yield a constant zero output for all inputs. This phenomenon was termed *dead ReLU* and described in [21]. Even though modern stochastic batch optimization methods mitigate this issue it triggered inventions of different parametric variations which aim to fix this problem (Fig. 1.10).



(a) ReLU activation function.

(b) Leaky ReLU ($a = 0.2$) and Exponential linear unit ($\alpha = 0.5$).

Figure 1.10: ReLU (a) and two of its parametric variants (b) that avoid the flat part and its negative effects.

Optimisation process

Formally, let $\mathcal{X}$ be an input feature space and $\mathcal{Y}$ be an output label space. The standard process of fully supervised learning involves learning a map $\mathcal{X} \mapsto \mathcal{Y}$ from a set of n labelled pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathcal{X}$ is a training sample and $y_i \in \mathcal{Y}$ is a label associated with $\mathbf{x}_i$. Training set is then a finite set $\mathcal{T} \subset \mathcal{X}$ of observations $\mathbf{x_i}$ and, accordingly, the learning phase is called training.

Among many different optimisation methods which are studied in the context of neural networks, first-order gradient-based methods are currently the most popular. These algorithms iteratively minimize a criterion function by taking proportional steps opposite to the gradient in a given point. Modern training schedules are very diverse, but in general a single iteration of the training loop consists of the following sequence of actions:

1. A sample input is processed by the model yielding an output.

2. The output is compared to the sample output and an error is measured using a predefined criterion function. The error and the criterion are called loss and loss function respectively.

3. The loss is differentiated with respect to all the network parameters and the parameter gradients are obtained. This done efficiently using the back-propagation algorithm.

4. The gradients are used to adjust the parameters under a chosen optimization schedule. The adjustment depends on the chosen optimizer and other involved regularization techniques.

The backpropagation algorithm computes partial derivatives of the loss function with respect to all network weights by repeatedly applying the chain rule transferring the network from the back to the front one layer at a time.

A loss function $\mathcal{L}$ is an arbitrary function differentiable with respect to the network parameters. In classification tasks, the output of the model is usually a discrete probability distribution over the domain classes C. To be able to measure the error of the output distribution, the true class label is also represented as a categorical one-hot distribution over the classes with one assigned at the index of the true label and zeros elsewhere. Then we can measure the difference between the two distributions by a crossentropy function defined as

$$\mathcal{L}_{CE}(\hat{y}, y) = \sum_{i \in C} -y_i \log\left(\hat{y}_i\right),$$

where for $i \in C$, y_i and $\hat{y}_i$ are the i-th elements of the distributions y and $\hat{y}$ respectively. A binary crossentropy loss l_{BCE} is a special case of crossentropy loss used in two-class classification problems and has the following formula

$$\mathcal{L}_{BCE}(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})).$$

Let $f(\boldsymbol{x}; \boldsymbol{\theta}), y)$ be a model output computed with parameters $\boldsymbol{\theta}$ and let $\mathcal{L}$ be a loss function. Then

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\mathcal{T}}}\ \mathcal{L}(f(\boldsymbol{x}; \boldsymbol{\theta}), y)$$

is an expectation of the loss over labeled samples $(\mathbf{x}, y)$ from an estimated training data distribution $\hat{p}_{\mathcal{T}}$. Since we aim to minimize the loss, we want to find $\boldsymbol{\theta}^*$ such that

$$\boldsymbol{\theta^*} = \arg\min J(\boldsymbol{\theta}).$$

Gradient methods allow us to approximate $\boldsymbol{\theta^*}$ by iteratively subtracting a scaled gradient of J with respect to parameters $J(\boldsymbol{\theta})$, denoted

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}),$$

where $\alpha < 1$ is called learning rate.

Following list presents three basic gradient descent variants used for neural network optimisation:

- *Batch gradient descent* calculates the criterion $J(\boldsymbol{\theta})$ using the whole training set $\mathcal{T}$ of cardinality M as

$$J(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right). \tag{1.2}$$

  This might be computationally infeasible for larger datasets of high - dimensional data due to memory requirements. The gradient then follows the actual sharpest descent towards the local or global minimum of the criterion function.

- *Stochastic gradient descent* (SGD), uses only a single random sample from $\mathcal{T}$ to approximate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. This estimate is naturally quite noisy, therefore the gradients often follow suboptimal directions. However, for simpler problems with a smoother function landscape, SGD offers probabilistic guarantees of convergence.

- *Minibatch SGD* combines the best of two previous methods as it uses a small random subset of m samples from $\mathcal{T}$ called minibatch in equation 1.2 to better estimate the true gradient. There is a trade-off between the computational cost and the accuracy of the estimate which is adjustable by a hyperparameter controlling the size of the minibatches. This option is usually the preferred approach to neural network optimisation.

1.2.3 Convolutional neural networks

In computer vision, the second decade of this century was dominated by deep convolutional networks (CNN). In the past decade, different variations of CNNs ultimately surpassed other alternative approaches and have become the model of choice for various image processing tasks. Convolutions have proven to be a superior tool for pattern recognition in domains of high-dimensional data with spatial relationships such as audio signal, time series, images and videos.

History

At the turn of the millennium a team of scientists led by Yann LeCun – a computer vision pioneer and one of the most respected reasearches in the field – designed and deployed a multi-layer convolutional network named *LeNet* to recognize hand-written characters on cheques. This was an unusual method at the time, though it achieved a record performance and was reportedly widely used in commercial sector in US and Europe. In his speech at CERN colloquium in 2016, LeCun stated that by estimation $10 - 20\%$ of all US checks were processed by his system in the early 2000s [22].

As a lifelong machine learning practitioner and a strong advocate of convolutional networks, LeCun successfully applied CNNs in multiple consecutive projects throughout his scientific career tackling some of the more challenging tasks of computer vision like face detection[23], object detection [24] and image segmentation [25]. All of these methods incorporated convolutions and achieved comparable results to the then state-of-the-art performance or even improved it in some cases. A hand-written ZIP code recognition system from 1989 [26] was one of the first demonstrations of the power of convolutions in pattern recognition tasks.

Structure

CNNs were designed to specifically address issues that make fully connected neural networks unsuitable for dealing with structured high-dimensional data.

First, if all pixels of a three-channel RGB image are used in an input vector, the number of network parameters becomes infeasibly large and the complexity of the network becomes unmanageable.

Secondly, fully connected neurons are not able to reliably encode local interactions between semantically connected input signals. In case of images, depicted objects usually have a coherent structure given by neighboring pixels of a localized region. Connecting every pixel of an input image to every neuron of the first fully connected layer disregards this input property.

Lastly, if one wants to detect patterns in an input with spatial or temporal dimensions, it is reasonable to expect that the patterns travel across the input.

All these these issues are addressed by structural aspects of a CNN resulting in the following properties:

1. Parameter sharing
2. Local interaction filters
3. Translation or shift invariance

In order to explain these concepts, we must first describe the mechanism the specialized CNN layers. Since many modifications and variations exist, we only provide a description of these concepts and structural principles of a basic CNN used in simple image recognition tasks. Therefore in the following text, words *input* and *image* might be considered equivalent even though the input domain is not restricted to images as mentioned earlier.

There are three main types of layers in a common CNN model:

- Convolutional layers performing feature extraction usually followed by ReLU activations
- Pooling layers subsampling tensors processed by the network
- Trailing fully connected layers which perform a classification over features provided by the preceding layers

A convolutional layer consists of a given number of kernels or filters which slide over the input in a predefined fashion and calculate cross-correlation in each position. A filter is a small rectangular region which simulates a receptive field usually in a form of a 3×3 or 5×5 matrix of trainable weight values and an optional bias. Cross-correlation is a mathematical operation related to convolution which, in a discrete case, can be viewed as a dot product of input tensor and the sliding tensor, the filter in this case. A single output of cross-correlation between a kernel and an input volume is a scalar value which represents a measure of similarity between the kernel pattern and patterns in the image positioned on different places in the input. As a kernel slides over the input volume it produces an output called activation map or feature map, which represents an activation of its pattern on different positions in the input. This process is also called filtering and addresses the local interactions between input values.

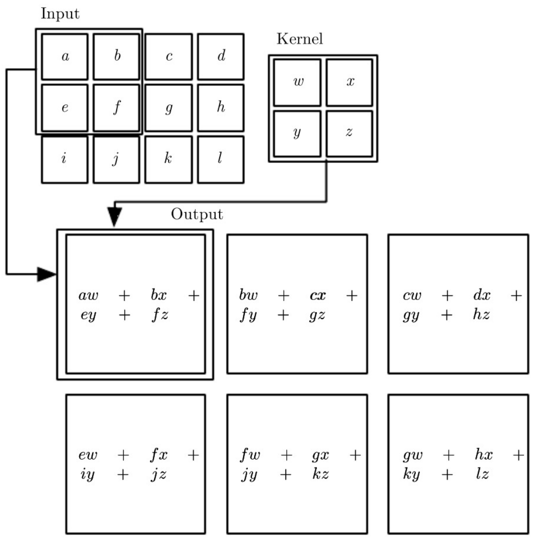
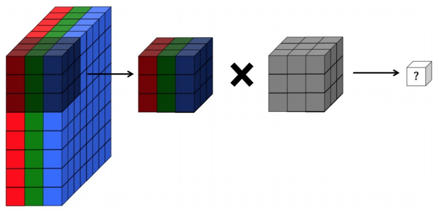


Figure 1.11: The calculation of matrix dot product or cross-correlation on a two-dimensional input. Source: [27].

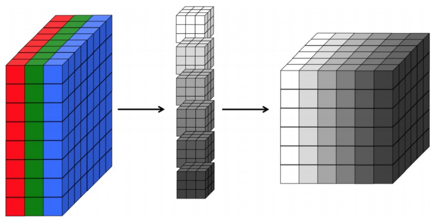
Although the operation is theoretically applicable on inputs of arbitrary dimensionality, in practice 1D, 2D and 3D convolutional layers prevail as they are sufficient for most tasks. A demonstrative example of calculation of a 2D cross-correlation is shown on figure 1.11.

The structure and behaviour of convolutional layer is subject to a rich set of hyperparameters defined prior to the training by the user, such as *stride*, *kernel size* or *padding*. The same hyperparameters also apply to a pooling layer described below. The role of different hyperparameters used in training and design of neural networks is further described in 1.2.4.

Every kernel of a convolutional layer is responsible for filtering the input for different pattern. In a fully supervised setting these patterns are gradually learned by a standard training processes described in 1.2.2.



(a) The $\times$ operation denotes cross-correlation of the gray kernel volume with an input frame volume in a single filtering step.



(b) Each layer of the activation map volume on the figure is a result of the filtering process with a kernel of related color.

Figure 1.12: An illustration highlighting layers of convolutional kernel volume used on a 3-channel RGB image (a) and a final activation map volume (b). Source: [28].

Parameter sharing is another important property of a convolutional layer which solves the problem of the network complexity by restricting every filter to a given set of weights. For illustration, a fully connected layer accepting a high-dimensional input like a one-channel image of $N \times N$ pixels would contain N^2 weights for every neuron. Conversely, a convolutional layer with parameter sharing and a single 3×3 kernel only contains 9 weights regardless of the input dimensionality. Refer to a schematic depiction on figure 1.13. An intuitive reason behind this technique is that patterns in the input image are not bound to their positions, which means the filter will register the same activation for a given pattern regardless of its position in the input volume. This dramatic parameter reduction helps to avoid overfitting and supports generalization performance.

Creating convolutional layers with a high number of kernels is desirable since it increases the capacity of the network, i.e. more filters are able to recognize more input patterns or their high-level combinations. This however increases the input depth linearly, therefore pooling layers were introduced to counterweight the effect and keep the input size manageable. The role of the pooling layer in a CNN is to reduce a spatial resolution, more specifically to decrease width and height of its input by merging regions of input values. The operation is very similar to the previously described filtering in a convolutional layer since pooling

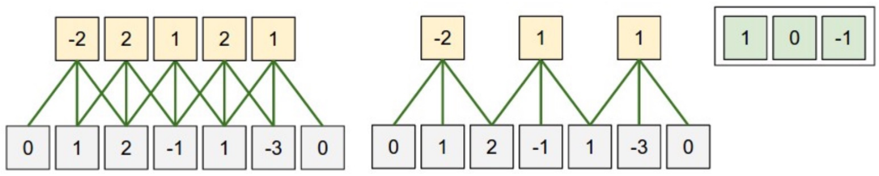
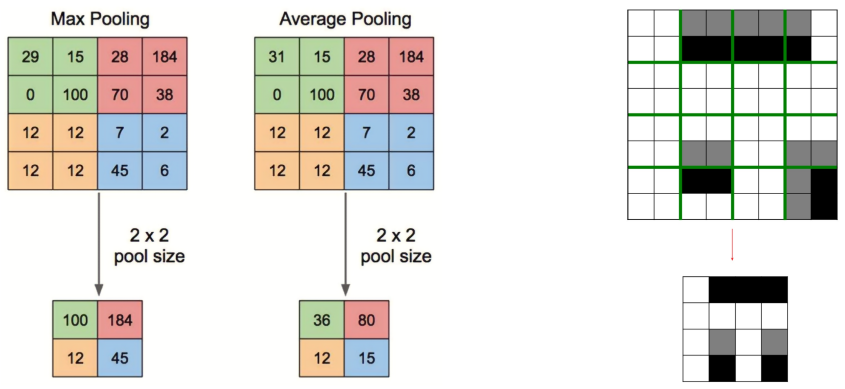


Figure 1.13: An illustration of the filtering process and weight sharing in a case of one-dimensional input with strides of 1 (left) and 3 (middle). The kernel weights (right) are shared through all of the filtering steps. Source: [29].

layers also produce outputs by using a fixed-shape pooling window sliding across the input. They usually do not contain any trainable parameters though and they only perform very simple operations. Instead of cross-correlation they typically compute maximum or average of the values in their current pooling window (Fig. 1.14).



(a) Max-pooling and average-pooling are two popular methods of pooling in common CNNs. Depicted are kernels of size 2×2 and stride 2. Source:[30].

(b) Pooling operators increase the tolerance of a network to input distortions and widen the spatial extent of subsequent filtering layers. Source: [28].

Figure 1.14: A pooling calculation executed by two popular pooling operators (a) and a schematic illustration of pooling distortion tolerance.

Parameter sharing and pooling operations together contribute to one of the most important characteristics of CNNs that is translation invariance. It is a property of mathematical objects and has connotations in geometry, but in the context of convolutional neural networks it means that the internal state of the network does not depend on positions of patterns in the input. This is an essential feature for a successful object recognition, since we expect an output to depend only the presence of the pattern only, not on its placement.

Note that shift invariance does not imply the existence of other invariances though. For instance, rotation, scale or flip invariances are also important but they are not guaranteed by default. If these properties are demanded, they are usually achieved by techniques of data augmentation, although it requires an increased network complexity to encode the broader space of internal representations. Alternatively, specialized CNN variants designed to address these shortcomings were proposed [31, 32].

Nowadays a common approach to design of a convolutional network purposed for classification is to treat the convolutional layer, activation layer and pooling layer as a coherent structural block. These blocks are then stacked on top of each other and finally, depending on the use case, appended with a few fully connected layers as shown of figure 1.15. Note, that due to inconsistent terminology, convolutional layer sometimes refers to this block of compound layers [27].

This cascade structure altering the feature extraction and pooling allows layers closer to the input to learn low-level features e.g. lines and edges, and farther layers to learn more abstract high-level features like shapes and textures [33].

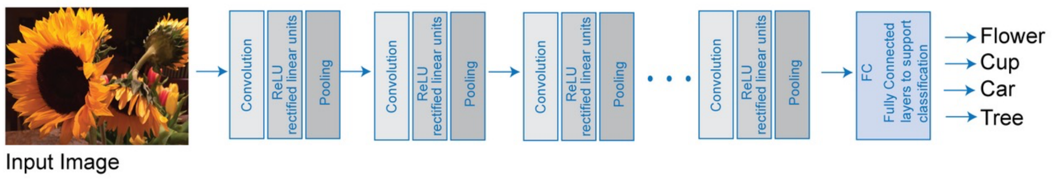

Input Image
Convolution
ReLU rectified linear units
Pooling
FC
Fully Connected layers to support classification
Flower
Cup
Car
Tree

Figure 1.15: A schematic illustration of the block structure in architecture of a deep convolutional network image classifier. Source: [34].

The input volume changes its spatial dimensions depending on the filtering and pooling settings in the network. The kernel stride and size affect the width and height of the input whereas number of kernels in a convolutional layer defines the depth of the input. The shape of convolutional layer output is determined by the following formulas:

$$W_{out} = \frac{W_{in} - F + 2P}{S} + 1,$$

$$H_{out} = \frac{H_{in} - F + 2P}{S} + 1,$$

where W_{in}, W_{out} and H_{in}, H_{out} denote widths and heights of input and output respectively. F, S and P represent hyperparameters of the layer, specifically kernel size, stride and padding. A demonstration of the effects of each CNN layer on the input shape is shown on figure 1.16.

ImageNet era

ImageNet is the largest online public database of annotated images. It was created in late 2000s by Fei-Fei et al.[36] and it shares some features with WordNet,

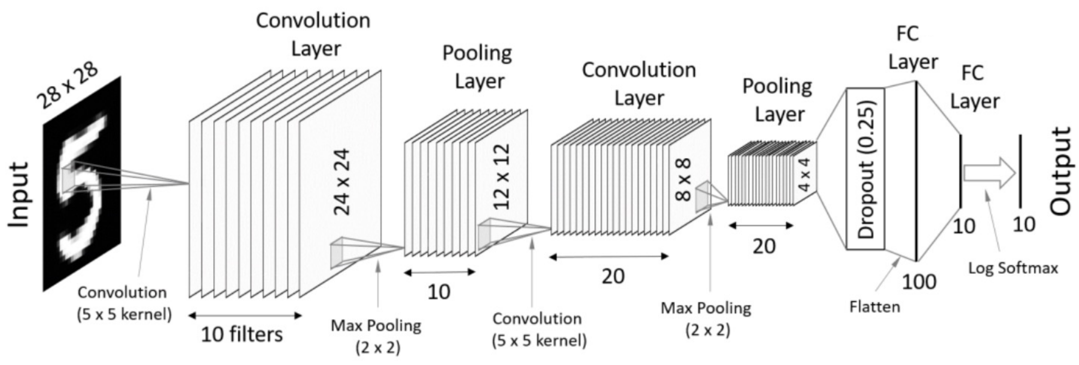


Figure 1.16: A simple CNN architecture used for recognition of handwritten digits. Similar convolutional networks gradually decrease width and height of the input in pooling layers to counterweight the increase of its depth in convolutional layers. Source: [35].

another public database, which reportedly served as an inspiration for the project in terms of hierarchical structure of the annotations. Nowadays ImageNet stores millions of images categorized into several thousand classes.

Between 2010 and 2017, ImageNet hosted an annual image recognition competition called *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) where contestants designed and submitted increasingly complex models every year in order to surpass the competitors while pushing the boundaries of the field. Its popularity quickly increased and within a first few years it became respected as a benchmark for measuring performance of state-of-the-art image classifiers from world's top research teams.

Popularization

There were a few notable milestones in the relatively short history of ILSVRC. Probably the most famous contribution was submitted in 2012 by Krizhnevsky et al. who proposed a novel approach to the image classification task by using a deep convolutional neural network commonly referred to as *AlexNet* [37]. The team won by a substantial margin and marked the beginning of the new era for the ILSVRC competition and for the whole image recognition field. In just two years after the AlexNet submission a majority of competitors shifted their focus to CNNs [22] abandoning all alternative methods.

Since 2012, every single one of the winning ILSVRC models used a CNN with some tweaks and improvements allowing for deeper architecture, faster convergence or better generalization. This vigorous research of increasingly deeper neural networks was supported by newly emerging technological abilities in terms of both software and hardware. As business gained interest noticed this phenomenon and gained interest in it, many robust optimised deep learning frameworks and libraries like TensorFlow and PyTorch were developed facilitating the whole design, training and deployment process. Furthermore, utilization of a specialized hardware optimised for fast matrix calculations like graphic processing units (GPU) and later tensor processing units (TPU) was a breakthrough innovation which generally sped up the training by a few orders allowing teams to train even very deep architectures with hundreds of layers in reasonable time horizons.

To illustrate the remarkable pace of the research powered by the ImageNet competitions, let us consider two winning models just three years apart. The AlexNet CNN in 2012 featured 8 layers whereas the ResNet CNN in 2015 already contained over 150 layers and the respective top-5 errors dropped from around 16% to below 4%. The estimated human recognition rate of 5.1% [38] was already surpassed by submitted models in 2015 [39], yet the accuracy of the winner models in the following years kept improving to the point where the classification task was pronounced solved (Fig. 1.17. The last ILSVRC competition was hosted in 2017 where the winning model achieved 2.251% top-5 error rate [40], less than a half error of the human error rate.

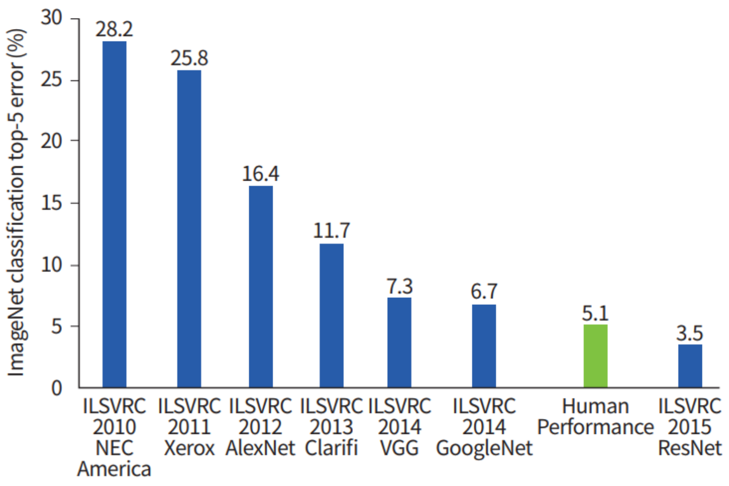


Figure 1.17: A graph showing the top-5 error development of ILSVRC winner models from the pre-convolutional era, over AlexNet in 2012 to ResNet in 2015. Source: [41].

Present state

The success of AlexNet was so significant it attracted a massive attention to neural networks in general. Over the following years, not only image recognition but also other research areas reported major leaps in progress by employing deep neural networks. A notable example of a field revolutionized by deep learning is natural language processing (NLP) which deals with speech recognition, machine translation, text generation, sentiment analysis and more. Other fields also benefited from the rediscovered potential of neural networks. Sophisticated image perception tasks like video object recognition, tracking and segmentation has seen a dramatic improvement with this approach as well. Reinforcement learning adopted neural networks and merged them with the established algorithms. The result of this fusion was popularized under the name *deep Q network* an originated from experiments by Mnih et al. They combined the Q-learning algorithm with a deep convolutional neural network to create a model which learns to plays a several old-school Atari games only from observations of raw pixel values of the game scene and a reward signal [42, 43], reaching a remarkable ability to learn the game rules and mechanics. A similar fusion pattern was also used in another expert-level game systems named AlphaGo and AlphaZero. These algorithms were optimised to play the game of Go on a superhuman level and they were able to beat the world champion for the first time in 2016 [44]. The game of Go was previously dominated by humans due to its enormous state space which was too complex for formerly used algorithms. Consequently, a whole new subfield termed deep reinforcement learning emerged studying the role of deep neural networks in reinforcement learning settings.

Recent skepticism

The role of complex CNNs in large-scale industrial cases is becoming increasingly important, which also creates a concern about their reliability. As mentioned, complex neural networks suffer from poor interpretability, which means it is difficult to understand the specifics of the input-output relationship they represent. This might pose a security risk in case of an attack exploiting its vulnerabilities. In fact, an adversarial attacks on AI systems in general have become a serious research topic in recent years. In terms of convolutional networks, the previous decade of optimistic eager research and successful applications is being slowly diluted by a certain level of skepticism. Recently, it has been revealed that even though current state-of-the-art image recognition models, which are based on very deep CNNs, achieve superhuman accuracy in certain image recognition tasks, they are shockingly vulnerable to adversarial attacks and exhibit a serious lack of robustness [45, 46, 47]

1.2.4 Deep learning

Deep learning is a subfield of machine learning studying optimisation techniques, design and architecture of neural networks with multiple hidden layers. The goal of deep learning is to learn complex networks from large quantities of data. The training process is computationally

Popularization

Even though deep learning gained its current popularity mostly in the last decade, the origins of the subfield date back to the previous century, notably to Y. LeCun and his research of convolutional neural networks thoroughly described in 1.2.3. The major factor which enabled application of deep learning in a large scale was mostly technical progress like increased computing power, specialized hardware and data availability.

There are two major properties that render deep neural networks successful nowadays. First, despite being generally very data demanding, the more data is supplied to the learning system, the better understanding of the concept is learned by the model and also the larger, more complex architecture can be trained. Secondly, deep neural networks are flexible and highly versatile autonomous feature extractors for a broad range of tasks if provided with enough computing power. This is especially useful in tasks where features are not provided and therefore are left to be extracted by the learner from the raw data.

Nowadays world's largest tech companies like Facebook, Google or Apple collect enormous amounts of data from their clients, which in combination with vast resources creates a perfect setting for deep learning methods of knowledge extraction. Many of real-world business applications were built using deep learning and neural networks [48, 49], although the specifics of these systems are naturally subject to trade secret.

Common issues

The structure and dynamics of deep networks bring along several complications which are not so pronounced in shallower networks. Apart from the lack of training data and computational power, which inhibited deep learning research in the past, there were also several problems related to the mathematical background. The rest of this section is dedicated to some of them, which are still relevant today.

Vanishing gradient One of the main issues which complicated training of deep networks in the past was presented by S. Hochreiter in his thesis from 1991 (original paper in German: [50]). The phenomenon was termed vanishing gradient and it became increasingly important with the current renaissance of deep learning.

The backpropagation algorithm calculates gradients of all trainable parameters by applying the chain rule, which results in repeated multiplication of gradients. As shown on figure 1.9, gradient values of common activation functions are always lower than one. Depending on the number of layers, tens or hundreds of small values are multiplied together including the final multiplication by learning rate which is also a small decimal number. The resultant gradients diminish exponentially with distance from the last layer causing the subsequent weight updates insignificant. In such cases, the training becomes inhibited or stops completely.

A variety of solutions have been proposed, some of the more recent ones recommend utilization of the following techniques:

- *Non-saturating activation functions*
  Different variations of ReLU have widely replaced the previously used activations as described in section 1.2.2. To reiterate, the main benefit of ReLU is its constant gradient of value one in the positive domain.

- *Proper weight initialization*
  Empirical tests have shown that an improper initialization of network weights results in deformed distributions of gradients in different layers, leading to reduced convergence rates and consequently poor final performance. Popular solutions of comparable results addressing this issue were proposed by Glorot and Bengion in [51] and He et al. in [52]. Their techniques have been quickly accepted as the best practice and implemented in all major deep learning frameworks.

- *Normalization of inputs*
  Input data features often come in different ranges which might complicate the learning process and hurt convergence in some cases. Normalization of the features is a cheap common practice to bring the feature values to comparable scales without any adverse effects. Depending on the data distribution, it is usually done by transformation to a standard normal distribution $\mathcal{N}(0, 1)$.

- *Batch normalization*
  Normalization across tensor batches commonly referred to as batch normalization has proved to have a remarkably positive effects on convergence rates especially for very deep networks [53]. The role of batch normalization as a regularization technique is described further in this section.

- *Residual connections*
  Residual connections (Fig. 1.18) are the building block of residual networks, a deep neural networks with modified connections enhancing gradient flow. First introduced in ResNet model [54], the winner of ILSVRC 2015, it has become a recognized architectural improvement for very deep networks.

These techniques contributed significantly to the present state of deep learning as one of the most versatile machine learning techniques and state-of-the-art solution for a wide range of tasks.

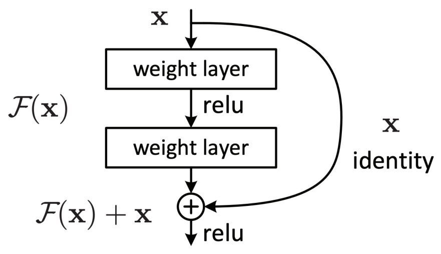


Figure 1.18: An illustration of the residual block structure. The identity connection helps to increase gradient values during backpropagation. Source: [54]

Hyperparameter space The rapid development of deep learning in recent years has resulted in additional complexity of the field. More specifically, the space of hyperparameters which should be considered in designing and training a model has grown substantially. Hyperparameters represent a family of decisions made by a human designer that are crucial for the output quality and usually cannot be altered ex-post without repeating the training process.

The following list suggests three categories of hyperparameters with selected representative examples based on their role in the development process and their impact on the model.

- *Architectural*
  These hyperparameters define the structural of the model network in terms of shape, depth and complexity. This category includes number of layers, number of units in each layer, activation functions, residual connections or loss function. Additionally in case of convolutional networks, number of kernels, kernel sizes, stride and padding (similarly for pooling layers).

- *Structural*
  These parameters are related to the first category but provide more subtle enhancements to the convergence speed or performance. These include dropout [55], weight size constraints, weight initialization, batch normalization and more.

- *Training-related*
  Decisions affecting mostly the training properties like batch size, number of epochs, weight decay, optimizer settings, scheduler settings, early stopping, label smoothing etc.

- *Data-related*
  Training set to test set split ratio, data normalization and data augmentations (flip, crop, cutout, noise, etc.) in case of image classification.

Note that the decision in majority of these options is not binary since most of those include their own set hyperparameters.

While some of these hyperparameters have a well-explained impact on the final outcome, in many cases it is very difficult to predict which combinations would achieve the best results. Therefore a common practice in deep learning project is to select just a small subset, repeat training for a few values and leave the rest on fixed values. This method is called hyperparameter grid search and common machine learning libraries like scikit provide optimized processes for that purpose. The ultimate goal of hyperparameter optimization is to improve generalization of the model. This approach is useful for models which do not require an expensive training like linear regression, support vector machine, k nearest neighbors etc. It is, however, problematic in case of neural networks since a training session might be, and usually is, very lengthy.

Choosing a suitable network architecture for a given task is essential for achieving a competitive performance. Overly complex models are computationally demanding in training and may even overfit to the training data. Conversely, simple models may not have enough capacity to learn the input domain appropriately. Designing a novel network architecture is a difficult process with no leading heuristics or guarantees of good results.

Neural architecture search

Designing a novel network architecture is very demanding in terms of time and computational capacity. The amount of hyperparameters and related design options has grown considerably with the recent increase in popularity and employment of deep learning. Hyperparameters and proper regularization are the key factors affecting training efficiency, size, robustness and overall performance of the final deep learning model.

Moreover, since the most common optimization technique is currently the stochastic gradient descent which offers no guarantees on resulting point of convergence, it is a common research practice to repeat the training several times and average the results since a single result could be noisy and not truly informative. This also greatly prolongs the whole research process and consumes even more resources. Especially when dealing with very deep networks containing hundreds of layers, a blind non-informed hyperparameter search becomes infeasible. Naturally a question arose whether the design process could be automated and optimized by some kind of heuristic. This was addressed by *neural architecture search* (NAS), a subfield of *AutoML*, which strives for automated network design using another AI techniques, often including reinforcement learning methods [56, 57, 58] but also evolutionary algorithms [59, 60] or combinations [61]. Architectures designed using NAS generally perform substantially better than human-designed models, currently the reinforcement learning NAS approach seems to be generally one of the most successful approaches as it often achieves state-of-the-art performance on a given task surpassing all previous rival traditionally developed models [62].

1.2.5 Model evaluation

Evaluation of a trained classifier is an important part of every machine learning system development. The purpose of evaluation in statistical ML is to empirically estimate generalization capabilities of the result model on a secluded testing dataset. Provided the testing dataset reliably represents the true data distribution of the task input domain, the result of the evaluation is then considered an accurate approximation of its future performance.

Different metrics are suitable for different cases depending on many factors like the type of the task (classification, regression or other), target application of the model, distribution of data classes etc. In the context of this thesis, we will focus on evaluation of classifier models.

In multiclass classification, for example in a common image recognition, accuracy is often a sufficient and widely used metric. Accuracy is the simplest metric measuring the ration of correct predictions to all predictions of the classifier. Image classification is a particularly simple example because in most cases the penalty for misclassification is uniform over classes. In other words, classifying a cat as a dog is considered no worse then confusing any other two classes.

Binary classifier evaluation

A binary classification task, also called dichotomy, is a special case of a multiclass classification which assigns an input one of two possible categories. Medical testing and spam filters are typical cases where binary classification is employed, as patients are either tested positive or negative for a given disease and an every e-mail is either spam or ham. These tasks naturally deal with heavily unbalanced class distribution in the dataset, which must be addressed during evaluation. The proportion of positive cases in such dichotomies is called prevalence and is usually very low. Accuracy is almost never a reliable metric for dichotomies since in a reasonable case of 5% prevalence, an obviously incorrect classifier predicting negative class for every input achieves a misleadingly high accuracy of 95%.

In order to avoid this issue more informative measures are used, a common practice is creating a confusion matrix which summarizes all predictions made by the classifier which are segregated in four distinctive categories depending on the relationship between a prediction and a true condition of the data (tab. 1.3). The four categories are termed *True Positive*, *False Positive*,*True Negative* and *False Negative*. The two-word names indicate which kinds of prediction-reality relationship they represent – the first word implies whether the contained predictions are correct, the second word denotes the predicted class.

		True condition	
		Negative	Positive
Prediction	Negative	TN	FN
	Positive	FP	TP

Table 1.3: Confusion matrix contains exact counts of each type of error and correct prediction made by a classifier. These counts help to better understand the true performance.

The elements of the confusion matrix might also be expressed as joint probability measures as presented in table 1.4.

Unit metric	Joint probability
True positive (TP)	$p(\hat{y}=0, y=1\|\mathbf{x})$
False positive (FP)	$p(\hat{y}=1, y=0\|\mathbf{x})$
True negative (TN)	$p(\hat{y}=0, y=0\|\mathbf{x})$
False negative (FN)	$p(\hat{y}=0, y=1\|\mathbf{x})$

Table 1.4: Meanings of unit metrics expressed as joint probabilities. $\mathbf{x}, y$ represent the observation-label sample and $\hat{y}$ is the label prediction.

The four elemental categories in confusion matrix are essential for more complex derived metrics. Some of them are listed in table 1.5. The main distinguishing factor of these metrics is their dependence on the sample prevalence.

These compound scores also have a meaning in terms of probability, not joint but conditional, presented in table 1.6

Sensitivity and specificity are frequently used in medicine for drug testing or disease diagnosis, whereas precision and recall is preferred in realm of computer science.

Another widely used methods which aid a visual assessment of the evaluation results are Receiver Operating Characteristic curve (ROC). ROC is a graphical plot measuring specificity on the x axis versus sensitivity on the y axis for the binary classifier while its discrimination threshold is varied. To reiterate, a binary classifier outputs a single probability value which represents its confidence about the affiliation of the input to one of the two classes. To transform the probability

Metric name	Formula	Prevalence
True positive rate (TPR), Sensitivity, Recall	TP/(TP+FN)	Independent
True negative rate (TNR), Specificity	TN/(FP+TN)	Independent
False positive rate (FPR)	FP/(FP+TN)	Independent
Positive prediction value (PPV), Precision	TP/(TP+FP)	Dependent
Negative predictive value (NPV)	TN/(TN+FN)	Dependent

Table 1.5: Basic metrics derived from the confusion matrix

Compound metric	Conditional probability
True positive rate (TPR)	$p(\hat{y} = 1 \| y = 1, \mathbf{x})$
False positive rate (FPR)	$p(\hat{y} = 1 \| y = 0, \mathbf{x})$
Positive predictive value (PPV)	$p(y = 1 \| \hat{y} = 1, \mathbf{x})$
Negative predictive value (PPV)	$p(y = 0 \| \hat{y} = 0, \mathbf{x})$

Table 1.6: Compound scores and their respective meanings as conditional probabilities. Note that the first two cases are conditioned by the true labels while the last two are conditioned by the label predictions.

into a two-class categorization, a discrimination threshold must be defined which creates a decision boundary separating the two classes. ROC takes candidate threshold values ranging fro 0 to 1 and for each value calculates the True Positive Rate (TPR) and False Positive Rate (FPR). Finally, these are plotted in a graph such as the one shown on figure 1.19.

A scalar score related to ROC is called Area Under Curve (AUC) and as the name suggests, it represents the area under the ROC curve. The AUC values range from 0 to 1, a random-decision classifier has AUC of value 0.5, while 1 denotes a perfect classifier. AUC values lower than 0.5 are rather odd since they symbolize binary classifiers which make wrong predictions most of the time – in that case it would be reasonable to switch the output predictions. On figure 1.19 the area under ROC curve is highlighted in light blue.

The diagonal line connecting the origin with the top right corner of the graph represents a random guess classifier and consequently every curve above the line is understood as a better-than-random classification. Furthermore, the closer the ROC curve is to the top left corner, the better the classifier skill.

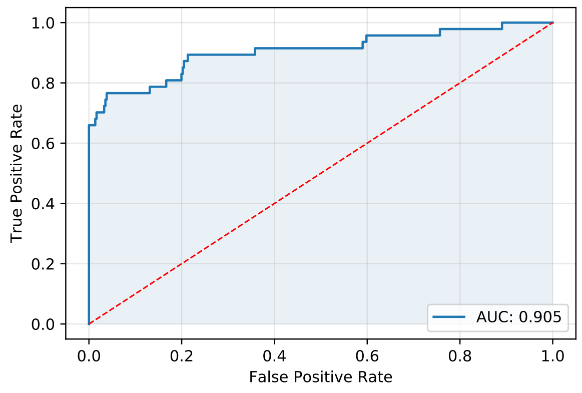


Figure 1.19: A ROC curve example with a highlighted area under curve.

1.2.6 Multiple instance learning

Multiple instance learning (MIL) is a special case of weakly supervised learning paradigm where instances of data samples are arranged in sets or multisets called bags. Ground truth labels are available only for the bags, instance labels may exist but they are unknown or hidden to the learner. The goal of MIL is to learn the underlying concept which correctly predicts the bag label given its unlabelled instances.

Formally, let $\mathcal{X}$ be an input feature space and $\mathcal{Y}$ be an output label space. The standart process of fully supervised learning involves learning a map $\mathcal{X} \mapsto \mathcal{Y}$ from a set of n training samples $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x_i} \in \mathcal{X}$ is a training sample and $y_i \in \mathcal{Y}$ is a label associated with $\mathbf{x_i}$.

In MIL, the training set $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_m, Y_m)\}$ consists of m labeled bags $X_i = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots \mathbf{x}_{i_k}\}$ and the goal is to learn a map $f_{MI} : 2^{\mathcal{X}} \mapsto \mathcal{Y}$

History

The term *multiple instance problem* was first defined by Diettrich et al. [63] in their drug research. The team measured bonding capabilities of drug molecules which may adopt several low-energy and high-energy shapes also called *conformations* determined by orientations of their bonds. Only some of these conformations bond well with a bonding site of some fixed target molecule hence the goal was to learn an instance-level binary classifier using labeled sets of conformations. Many variants of MIL exist today. Most of the work in the area is still done under assumption of binary labels for input instances, however more complex MIL concepts were also proposed [64].

Terminology

Although terminology in MIL is rather vague, there have been attempts to structure the methods and create a basic taxonomy and nomenclature. A notable contribution was made by Gärtner et al.[65] who proposed following definitions and terms which are commonly used in MIL research today.

Definition 1.2.1 (Concept). *A concept is a function $\nu_I : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is an instance space and $\mathcal{Y} = \{\perp, \top\}$ is a label space.*

A concept over an instance space is called *instance concept* or, according to Gärtner, an *underlying concept.*

Definition 1.2.2 (Multi-instance concept). *A multi-instance concept is a function $\nu_{MI} : 2^{\mathcal{X}} \mapsto \mathcal{Y}$ defined as:*

$$\nu_{MI}(X) \Leftrightarrow \exists\ x \in X : \nu_I(x),$$

where $X \subseteq \mathcal{X}$ and ν_I is an instance concept over $\mathcal{X}$.

Single-instance learning (SIL) is a special case of MIL where each bag contains exactly one instance. In this setting, the bag labels become instance labels and the learning process is analogous to classical fully supervised learning.

Assumptions

A bag label is determined in a certain way by labels of the instances. This is referred to as the *multiple-instance assumption*, which defines two basic categories of MIL tasks – a standard MIL and a generalized MIL. The original idea of this two-level hierarchy view of multiple instance was originally proposed by Wiedmann et al. in [66].

The standard MIL assumption states that a bag $\mathcal{B}$ is labeled positive if at least one of its instances is labeled positive. Similarly, a bag is labeled negative if all of its instances are labeled negative. A formal notation is as follows:

$$\mathcal{Y}_{\mathcal{I}} = 1 \Leftrightarrow \sum_{i \in I} [\![y_i > 0]\!] \geq 1$$

$$\mathcal{Y}_{\mathcal{I}} = -1 \Leftrightarrow \sum_{i \in I} [\![y_i > 0]\!] < 1$$

Complementary, the generalized MIL works with a threshold k which defines the minimum number of positive instances in a positive bag:

$$\mathcal{Y}_{\mathcal{I}} = 1 \Leftrightarrow \sum_{i \in I} [\![y_i > 0]\!] \geq k$$

$$\mathcal{Y}_{\mathcal{I}} = -1 \Leftrightarrow \sum_{i \in I} [\![y_i > 0]\!] < k$$

Based on these definitions, the standard MIL might be seen as a specialized case of the generalized MIL, where the threshold $k = 1$.

2 | Related work

Contents

2.1 Task-related **39**

2.2 Model-related **42**

This chapter reviews projects with goals similar to our task and and other related research. The first section provides an overview of work related to our task of automatic diagnosis of multiple myeloma, the second section then studies applications of convolutional neural networks on CT images in general.

2.1 Task-related

Here we provide an overview of work closely related to our task of automatic diagnosis of multiple myeloma. A proper description of the disease is given in 1.1.3 but to restate the problem, multiple myeloma is a malignant cancerous disease affecting plasma cells. One of the main characteristic symptoms of the disease in advanced stages is development of medullary osteolytic lesions. The lesions are clearly visible under computer tomography screening, which is usually a recommended diagnostic tool for this purpose. Currently a radiologist or other clinician is required to examine the images visually, which might be inefficient or insufficient if the examiner is unsure or still in training. An automated diagnosis of these images providing another diagnostic opinion to the human expert would be appreciated as an educational or practical tool.

Under screening, these lesions exhibit a different density than the surrounding bone marrow. Many approaches in this area exploit this fact to create a probabilistic models of marrow voxel intensities of healthy femurs an then detect the lesions as outliers of these models. An automatic diagnosis of multiple myeloma is interestingly not a frequent research topic, nevertheless we present several notable exceptions.

The most closely related research projects come from Martínez et al. [67, 68] and Hering et al. [69] who focused on automatic MM diagnosis from CT images of femurs. A similar problem was addressed by Omiotek et al. who worked with a dataset of computed radiography (CR) images of humeri. Following text describes their approaches in depth and highlights notable differences.

[67] and [68] are methodologically fairly similar subsequent projects with a few differences. Both deal with the variability of femur shapes in a rather interesting way. To unify the scale of all femurs they introduced a transformation

of the original (x, y, z) coordinates of marrow voxels to normalized radial and longitudinal $(l, t) \in [0, 1] \times [0, 1]$ coordinates (Fig. 2.1).

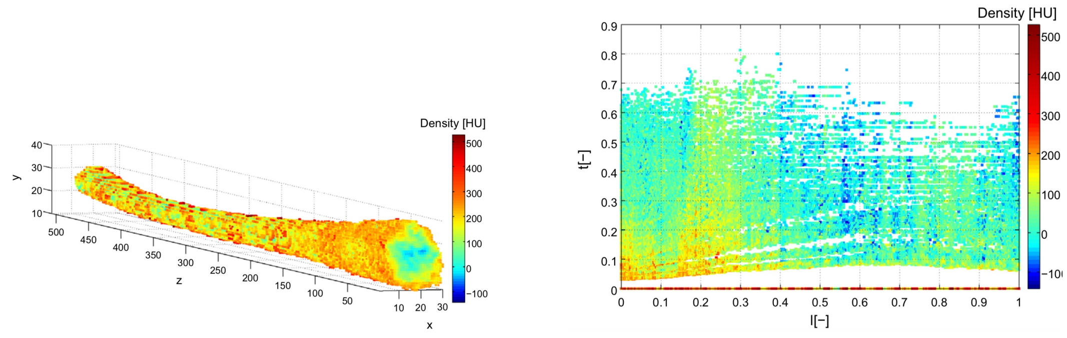


Figure 2.1: Radiodensities of bone marrow voxels. The Euclidean space (left) was transformed to the (l, t) space (right) for the purpose of unification of highly variable femur shapes. Source: [67].

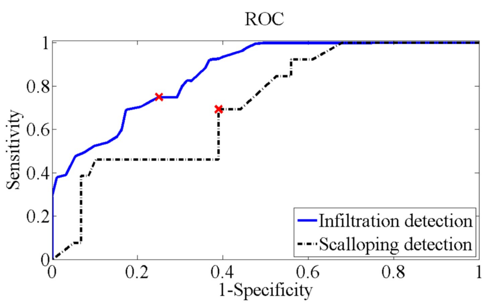
Authors of [67] focused on both infiltration detection and scalloping detection. They collected healthy femurs and created a probabilistic model of marrow voxel intensity values. To achieve this they split each femur to 100 sub-volumes and computed histograms of marrow intensities on each of them. Corresponding histograms were then summed across all femurs in a training set, normalized and rescaled to a fixed range. These histograms represent the probabilistic model, individual voxels in test images are then compared to these histograms and detected as outliers of the model if

$$p\left(I > I(l,t)\right) < \xi_0,$$

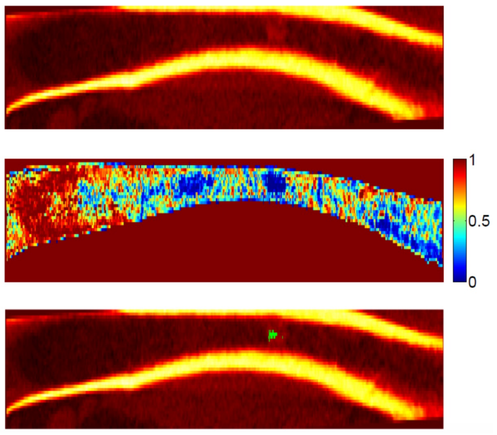
where $I(l, t)$ is the intensity of the voxel with coordinates l, t. I is the intensity on that position in the corresponding histogram and ξ_0 is a parameter threshold. A demonstrative example of detection of infiltrated voxels in a diseased femur is shown on figure 2.2b.

The whole marrow mass is then considered infiltrated if it contains at least ρ connected components of τ outlier voxels. The three thresholds were varied to optimize performance which yielded the results presented on figure 2.2a.

This work was extended in [68] with a few additional improvements. First, the cortical scalloping detection was omitted probably due to unconvincing results. The diagnosis of MM is based on the presence of the marrow lesions anyway, the scallops are merely a secondary indicator and often less prominent. Also, to localize bone marrow the authors segmented a cortical bone by a simple intensity thresholding procedure. While this method reliably identifies most of the cortical bone mass, it is overly rough on the edges of the bone. Especially on the border between cortical bone and bone marrow, this method defines a strict border,



(a) A ROC curve showing lesion detection and scalloping detection of the final model with optimised thresholds ρ, τ and ξ_0.



(b) Original image of visibly infiltrated femur (top), probabilities for each marrow voxel according to the model (middle) and a detection of infiltrated voxels given selected thresholds (bottom).

Figure 2.2: Figures from [67] showing performance of the proposed model (a) and a demonstration of an infiltrated area detection (b).

which is not necessarily authentic and more importantly it introduces unwanted artifacts, which may resemble scalloping. Second, the ρ and τ thresholds were used as a two-dimensional feature vectors and k-nearest neighbors and support vector machine were employed to perform classification in this vector space. This approach lead to substantially better results exceeding 0.99 AUC.

Omiotek et al. [70] assessed a large variety of different machine learning and statistical methods suitable for diagnosis of the MM from marrow lesions. Instead of femurs, they focused on humeri and performed an extensive image preprocessing in order to extract descriptive features from the marrow. Note that their dataset consisted of computed radiography (CR) images unlike the other mentioned studies which worked with CT images. They claim to have identified 279 distinct bone marrow descriptors, which were then reduced to top ten most informative combinations of features using rather unknown Hellwig's method of feature selection [71]. This method supposedly finds the best feature combinations which exhibit the weakest intercorrelation and simultaneously the strongest correlation with the dependent variable (a diagnosis label in this case). A total number of 90 experiments were conducted – one for each combination of a classifier model and a feature set, which lead to a conclusion that in all cases an instance of k nearest neighbors classifier (k-NN) was dominant in terms of accuracy and binary classification metrics like *TPR*, *TNR*, *PPV* and *NPV*. Finally, they proposed and described a fully automatic CAD system for usage in clinical practice of medical screening.

To our knowledge, the most recent work related to the topic of this thesis was published by Hering et al. [69] who attempted to use generalized multiple instance learning (MIL) on the same dataset of labelled femurs for the purpose of localizing the marrow lesions. From a relatively small dataset of about 200 femurs, their method learned to infer locations of lesions in the bone marrow under certain assumptions from femur-level labels only without any further information about number or positions of the lesions (Fig. 2.3). Unlike [67] and [68] this approach does not build a probabilistic model of marrow intensities. The proposed method of feature extraction splits each femur to N_z cylindrical regions of interest (ROI) with N_t intervals of the radius and N_ϕ intervals of the angle coordinate. Each ROI is then appended with a vector of intensity features consisting of the mean, the standard deviation and a histogram with N_h bins. Finally, by concatenation of these features along with the coordinates of the cylindrical split, each ROI is assigned a feature vector

$$\mathbf{x}_i = [l, t, \phi, \mu, \sigma, h_1, \dots, h_{N_h}] \, ,$$

which represents i-th instance in a bag of instances. The authors conducted several experiments using random forest and SVM classifiers modified for learning from multiple instances. The results of the best performing model achieved a near perfect score of > 0.98 AUC.

The utilization of multiple instance learning in the area of automatic multiple myeloma diagnosis was a novel approach and no similar known method was found by the authors at the time.

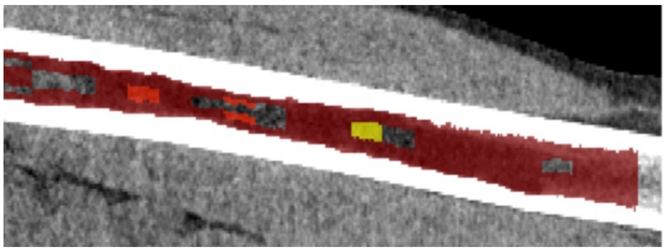


Figure 2.3: Marrow lesions localized by the MIL classifier proposed in [69]. Source: original paper.

2.2 Model-related

The research projects mentioned in the previous section often perform some kind of manual feature extraction or marrow segmentation. It is a reasonable approach to this specific task since the relevant part of the input, which carries the majority of the information (bone marrow), can be quite simply localized and extracted by a deterministic procedure. However, with the power of modern deep convolutional neural networks, it should be also feasible to leave the feature extraction to the model and learn from raw image data.

CNN are currently being applied in many different image recognition task across multiple fields. Medical imaging is no exception to this rule. Authors of [72] support this claim by their extensive survey of deep learning methods in medical image analysis, which revealed that the vast majority of over 300 participants used a CNN-based model in their research projects.

However, not much research is done specifically on CAD systems for multiple myeloma despite the wide occurance of the disease in the population. In the context of 3D medical imaging modalities like CT, MRI or PET, we find that pneumonia, tuberculosis and other pulmonary diseases seem to be much more prevalent research topics according to our analysis [73, 74, 75, 76, 77]. This might be attributed to the emergence of publicly available datasets and competitions even in the area of medical data. For instance, LUNA16 [78] is a project which for several years held a CAD competition on IDRI dataset [79] – a popular large-scale dataset of thoracic CT scans with manually annotated and segmented lung nodules.

The goal of all of these research projects is to perform a diagnosis in a form of binary classification on the CT or X-ray image inputs. Some of them also produce a segmentation maps of nodules if the chosen dataset supports it. A variation of 3D convolutional network was used in all of the cases, either for the classification or the segmentation purposes. Note that the following text omits the description of the exact achieved results, since the reported scores differ only marginally in a $> 90\%$ accuracy range. Most of the papers report results, which are on-par with state-of-the-art models or fairly similar. The main differences are observed in proposed network architecture designs, which are briefly mentioned here.

Authors of [77] employed a 3D modification of Faster R-CNN, which is a specialized neural network architecture widely used for image segmentation, and then classified the nodule regions using gradient boosted machine. On the contrary, [73] proposed an interesting original multi-task network architecture for a simultaneous segmentation and classification. Their motivation is to provide the classifier not only with the small segmented regions of interest, but also with contextual information from a broader input area.

Dou et al. [80] participated in the LUNA16 competition with a nodule classifier model aimed for reduction of false positive diagnoses. They proposed a novel classifier architecture consisting of an ensemble of three 3D CNNs each processing a different size of a candidate nodule region. Finally, an affine combination of the three output probabilities was calculated to obtain the final verdict.

Nowadays neural networks are also being applied to more traditional tasks like data compression, image/video colorization or image/video quality enhancement.

The last named category also includes super resolution, which is a method of artificially increasing resolution of an image. Reportedly, neural networks trained on large datasets of similar images have the ability to perform super resolution better than conventional methods. This is the case of Wang et al. [81] who trained a novel deep 3D CNN model to upsample resolution of input CT images and reported an increase in the output image quality compared to similar non-neural methods.

Our goal and dataset were most similar to those of Huang et al. in [76], who trained a reasonably deep 3D CNN binary classifier on a small dataset of only 99 lung CT images. They report state-of-the-art results. Their proposed architecture served as an inspiration for our classifier model.

To our best knowledge, there is no published work combining convolutional neural networks and multiple instance learning for the task of multiple myeloma diagnosis from medical images.

3 | Dataset

Contents

3.1 Structure 45
3.1.1 Original dataset 45
3.1.2 Derived dataset 46
3.2 Data cleaning 47
3.2.1 Femur skew correction 47
3.2.2 Bone implants 48
3.2.3 Removing joints 48
3.2.4 Removing air voxels 51
3.3 Data flow 52
3.3.1 Creating femur patches 53
3.3.2 Augmentations 54
3.3.3 Transformations 57

Unlike ordinary image datasets, acquiring annotated medical image data is quite problematic for multiple reasons. First, they are produced by obviously very expensive methods and usually in smaller counts. Second, the annotation procedure requires an assessment from medical experts, which is time consuming and also costly. Lastly, data about personal health are confidential and must be anonymized prior to their public exposition. The following sections describes the structure of our dataset of CT scans and operation taken in order to reshape it for the purpose of our thesis project.

3.1 Structure

This section describes the refinements done to the original dataset in order to extract information relevant for our task. The performed changes were quite significant, therefore we distinguish the original and the final transformed dataset here.

3.1.1 Original dataset

The original dataset available to us contained lower body CT images capturing both femurs of about 290 patients, which were diagnosed by professionals for presence or absence of multiple myeloma lesions. Each femur was diagnosed individually and annotated with a label according to the level of infiltration observed in the sample. The following table shows the categories of annotations which were used.

Annotation	Meaning
-1	Healthy patient
0	No visible infiltration, blood test positive for MM
1	Small infiltration visible
2	Medium infiltration visible
3	Large infiltration visible
4	Some infiltration visible (not quantified)

Table 3.1: The original dataset annotation structure.

The global label of a lower body CT image is created by concatenation of the labels of both femurs in the image. For example, label 23 stands for medium and large infiltration in left and right femur respectively in the given CT image. However, most often both femurs of a patient fall in the same category.

3.1.2 Derived dataset

For the purpose of this thesis, we utilized a dataset of cropped legs which was derived from the original dataset prior to this thesis. Moreover, the labels of the femurs were simplified to binary indicators of the disease, i.e. the scale of the infiltration is omitted in our experiments. The cropping process consisted of two phases:

1. Localization and segmentation of femurs by a thresholding procedure.
2. Creating a three dimensional cuboid encompassing the femur with a certain clearance. Note that in the following text the term *cuboid* will always address the notion defined here.

In addition to the images of cropped legs, we also generated dictionary files containing dimensions of bounding boxes for each layer of each femur. In the context of this thesis, *bounding box* is a rectangle circumscribed about the femur in its layer in axial view 3.4 separating it from the surrounding tissues. Bounding

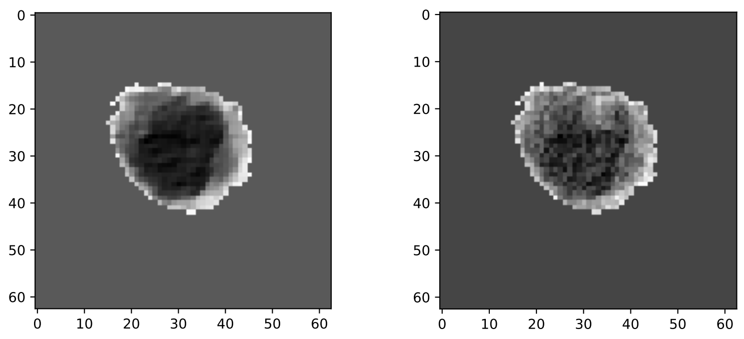


Figure 3.1: Axial view of a femur layer with segmented marrow mass without lesion augmentations (left) and with full augmentation (right).

boxes are important in a process of creating image crops closely focused on the femur. This process is described in section.

Lastly, we also created segmentation maps for each femur, allowing us to mask out three different parts of a femur image. Figure 3.1 displays the effect of each individual mask on a sample image. Refer to chapter 5 for a detailed description of the masking process.

3.2 Data cleaning

Data preparation is an integral part of every data mining and machine learning application. A related term *data cleaning* is a modern jargon which refers to a set of actions taken on the data for the purpose of reducing noise, removing inconsistency, detecting data outliers, correcting data class distributions etc. Cleanliness of the data is a crucial factor affecting the final outcome of the project and is therefore often dominant in overall time costs. Our dataset also contained several inconsistencies like outliers or irrelevant data, therefore we decided to perform a thorough cleaning before proceeding. The following subsections describe encountered obstacles and the respective actions taken to overcome them.

3.2.1 Femur skew correction

Since the vertical axes of the cropped femurs were not always aligned with the vertical axes of the their cropping cuboids, some femurs laid askew in their cuboids, which was the source of following inconveniences.

- Because of the skew of the femur its bounding cuboid needed to be unnecessarily large to encase the whole femur. This means that the relevant parts like the marrow would be relatively small compared to the unimportant surrounding tissues which will prevail in the cuboid.

- An excessive cuboid size also means it often spans outside the leg which increases a chance of containing unwanted air voxels.

The mentioned issues were fixed by rerunning the cropping procedure on the original dataset while aligning the vertical axis of each femur with the vertical axis of its cuboid. The principal vertical orientation of the femur was approximated by a straight line which passes in a minimal distance to the geometrical center points of bone marrow in the first, the middle and the last axial slice.

Femur skew was also tackled in the related work of Martínez et al. [67] mentioned in chapter 2, who used principal component analysis (PCA) to estimate the direction of the largest variance of bone marrow voxels and rotated the original images and masks by the angle difference.

3.2.2 Bone implants

Around 10% of the femurs in our dataset were found to have a metal implant placed in the bone. These femurs are problematic for our experiment because the dense implant material interferes with its surrounding tissues in the CT image, or it even replaces a part of the femur marrow completely as presented on figures 3.2 and 3.3. Since these implants are mostly (but not always) located at the end of the femur on the hip side, a reasonable idea would be to cut them off the femur images and leave only the rest. However, these implants often reach far in the femur and they appear gradually along the vertical axis in the femur image so choosing an appropriate spot to make the cut is not trivial. Algorithms that try to locate a suitable cutting spot were developed as a part of this project, although we eventually decided to remove these femurs from our dataset since the benefits of the correction procedures were too subtle and the cut femurs would not fit our experiment well for technical reasons.

Apart from the bone implants, some images also contain stents, a plastic or a metal tube inserted into a blood vessel which supports blood flow, or different unidentified metal objects located in fat or muscle tissues (Fig. 3.2, bottom). Unlike the bone implants, these objects did not require any corrective actions, since they were not obstructing the marrow and were distant enough to get cropped out later in our data preprocessing pipeline.

3.2.3 Removing joints

We aimed our focus on femurs in this project due to their long uniform shape and voluminous marrow mass, which provides better conditions for our classifier. To our knowledge, there is no evidence that the lesions form in some part of femurs more often than in others or that they follow some non-uniform distribution of localization. In serious advanced stages of multiple myeloma, the lesions are present and visible in the marrow of the femoral shaft as well as in the joint.

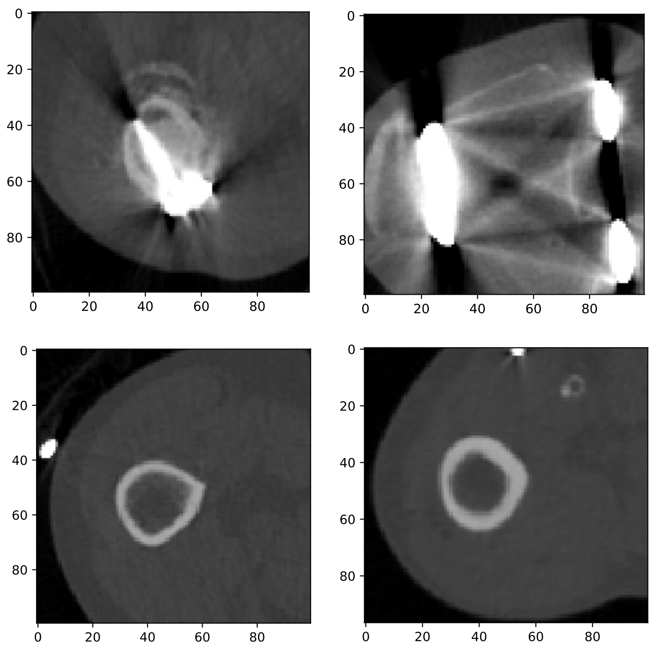


Figure 3.2: Axial view of various bone implants interfering with the femur bone marrow (top) and small unidentified metal bodies, presumably debris (bottom).

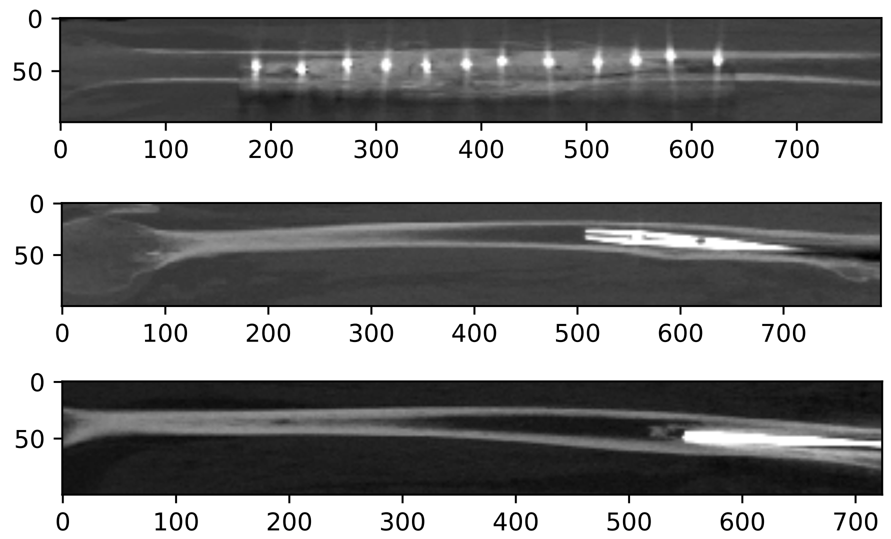


Figure 3.3: Longitudinal sections of femurs with different metal implants.

Our initial examinations uncovered that the edges of each cropped femur image in our dataset capture joints. These joint areas differ substantially from femur shaft areas as they tend to be much wider, they are strongly asymmetrical and their inner structure and density are also different since they contain a high-

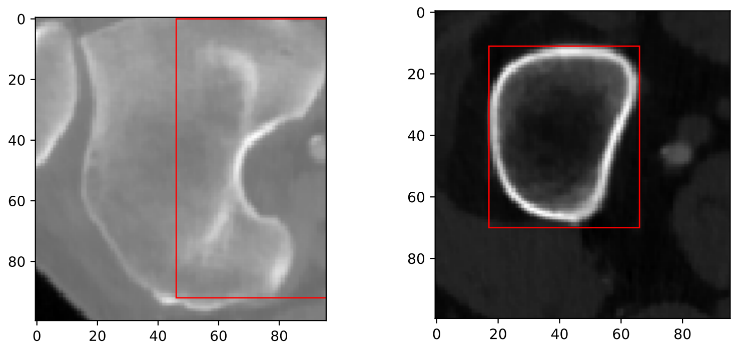


Figure 3.4: Axial section of a joint (left) and a femoral shaft (right) of the same femur sample with bounding boxes highlighted in red color. Note how the automatic bounding box generation procedure fails in the case of the joint.

density cancellous bone (Fig. 3.4).

Obtaining more data to fix this problem was not possible, so we agreed to cut off the joints from both ends of each femur. This does not sacrifice the generality of our task if our hypothesis about lesion distribution along the femur holds, i.e. if the disease is in a certain phase of severity, the whole body of the femur marrow including joints is affected evenly which means cutting off two relatively small parts from the femur should not change the overall diagnosis based on visual aspects.

The problem of cutting off the joints is not as straightforward as might seem though, mainly because one has to locate a suitable place to make the cut for each uniquely shaped femur. The task may be equivalently reformulated to locating a femur-joint border, which is obviously a very vague task definition and should illustrate the problems associated with it. After thorough discussion of possible approaches, we decided to use the size of bounding boxes of a femur as joint indicators since all joints contain much larger bounding boxes. The final solution was to locate the femur-joint border by calculating a layer-wise geometric means of width and height of all bounding boxes of a femur, taking their median and multiplying it by a constant factor. This yielded an upper limit value for the geometric mean of bounding box dimensions in all axial layers of the femur.

In some femur layers the procedure generating the bounding boxes fails to some extent, most often in joint areas as shown on figure 3.4. However, in some rather rare cases the bounding box is enlarged to encase a nearby tissue, which was mistaken for a bone by the detection algorithm (Fig. 3.5). Such poorly generated bounding boxes then cause spikes on the BB graphs, which must be addressed as shown on figure 3.6.

Martínez et al. in his related research [68] mentioned in chapter 2 also identified joints as problematic areas of increased density and performed correction by

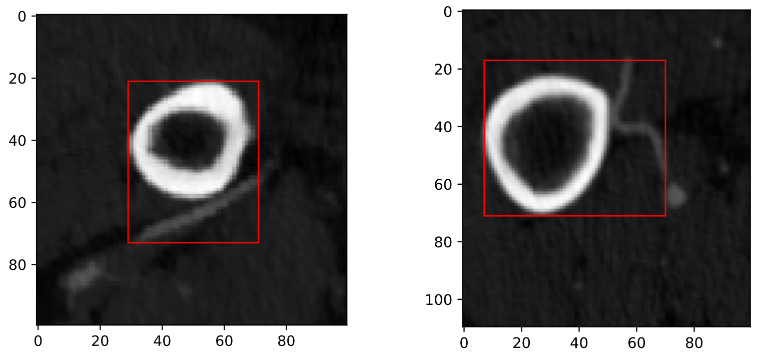


Figure 3.5: Bounding boxes erroneously encasing adjacent tissues of higher density.

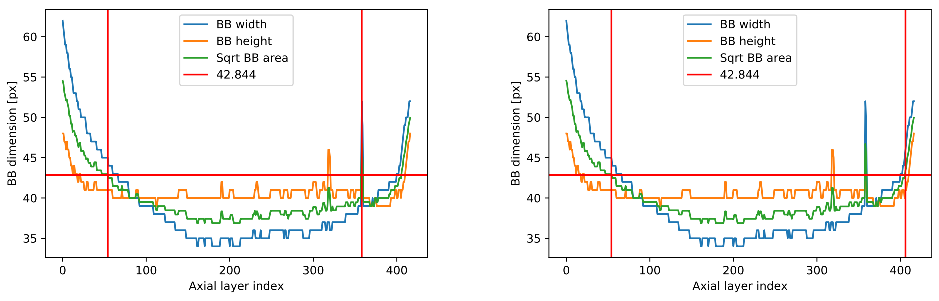


Figure 3.6: Graph of width, height and square root of the area of bounding boxes measured on each axial layer of a femur. Two different settings are presented here regarding the treatment of the graph spikes. The procedure can either halt on spike detection (top) or ignore it (bottom). The scaled median level is represented by the red horizontal line. In both cases, the median scale factor is set to 1.1. The vertical lines mark the found cutting spots.

removing 15% from both ends of the femur cuboid in the longitudinal (vertical) axis. This approach is rather simple and significantly less robust than our solution, which treats femurs individually, takes into account differences in structures and finds a unique cutting spot for each femur.

3.2.4 Removing air voxels

A CT screening naturally captures the surrounding environment of the scanned object, the air is present on the final image in the majority of cases. As presented in table 1.1, the air is projected to the lowest value of -1000HU on the CT number scale, which renders it pitch black after mapping to a greyscale color space (Fig. 3.7).

Depending on the thickness of a patient's leg, cropped images of legs in our

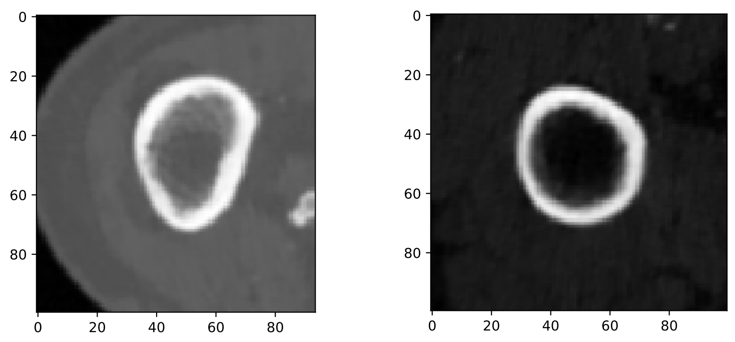


Figure 3.7: Axial view of femur crops with air voxels (left) and without air voxels (right). We aimed to remove the air voxels from all images in our dataset since this disparity in appearance creates an unwanted variance in the input distribution and might hinder the classifier training.

dataset contain air voxels, usually around the knee area, where the bounding cuboid spans outside the leg. The air is unwanted in the image, since it shifts ranges of intensity values after mapping to gray scale. This naturally leads to To illustrate the problem consider a normal CT image containing a femur and its surrounding tissues like muscles and fat. The intensities in such image range from around -200 HU to 1800 HU according to table 1.1. Conversely, if air voxels are present in the image, the lower limit of the range is extended to -1000 HU. The latter creates a much wider range which then needs to be mapped to only 256 grey steps ($[0, 255]$ value range of an 8-bit gray scale). In clinical practice, this very issue is addressed by the windowing technique described in section 1.1. Figure 3.7 provides example images of these two cases to illustrate the problem.

After a thorough discussion and different proposed solutions, we settled with a simple method of value clipping to remove the air voxels where all image patches are clipped to a $[-200, 1800]$ HU range. This interval was chosen as it created least visible edge between an outer leg and the air.

This issue is mostly present in our derived dataset described above. However, it becomes very rare after the second cropping based on femur bounding boxes, which is explained in the next section.

3.3 Data flow

Following subsections describe the operations and transformations applied on the raw data from the our dataset in order to obtain tensor batches which are compatible with the input requirements of our classifier network.

3.3.1 Creating femur patches

Since our model of choice is a convolutional neural network, its inputs must obey predefined shape requirements in order to get processed. Clearly, feeding the whole unprocessed CT images of cropped legs is infeasible, partly because of the memory requirements, but mainly because of the variability of their shapes as shown on figure 3.8. Obviously, without any deforming transformations, the depth of the cuboid encasing the leg varies greatly depending on the length of the patient's femur.

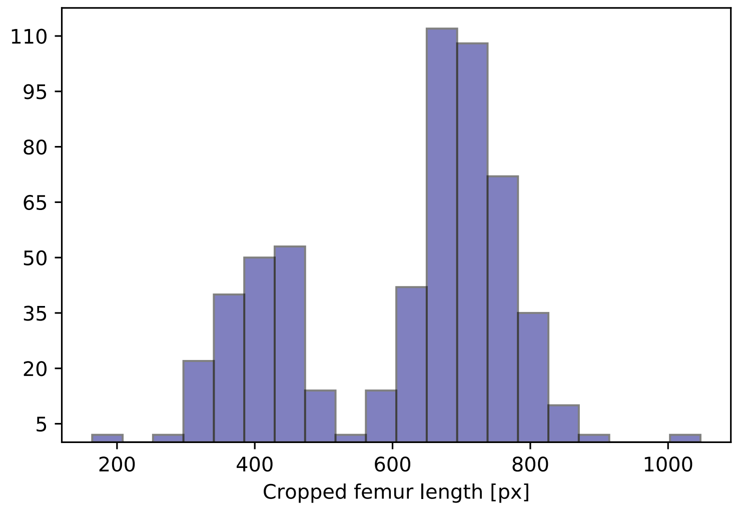


Figure 3.8: Histogram showing the variance in lengths of cropped femurs with removed joints in our dataset measured in pixels.

Our solution to this is to split each femur along its vertical axis in a set of smaller *patches* with a fixed height. Each patch also overlaps the previous patch by an adjustable constant number of voxels. Throughout the rest of this text, the term patch will always refer to a chunk of femur CT image as defined here.

After patching the femur, the remaining two dimensions of every patch also needed to be brought to some fixed values for the same reason. These values were unknown and we examined two different approaches in order to determine them.

The first attempted approach was to rescale both width and depth of every patch to a predefined value. This is a straightforward solution, although it has its disadvantages. Firstly, if the amount of useful information in the patch, which is mainly the bone marrow, is already relatively small compared to the whole patch, downscaling along any axis makes the valuable information even smaller which means it will be more difficult to diagnose and classify. On the other hand, upscaling by any axis means extrapolation on values in that direction and creates artifacts in the image which might also confuse the classifier considering the degree of the rescaling in both axes is variable across all patches.

Therefore, a more sophisticated method was developed using the bounding boxes mentioned in the previous section which frame the area of the femur in each of its layers along the vertical axis. Since our task focuses on multiple myeloma lesions present in a bone marrow, the surroinding tissues around the femur are rather distractive to the classifier. The goal of this idea is to create a tight crop of the femur removing as much of the irrelevant tissues as possible. In other words, minimizing both dimensions of the bounding box which is needed to perform the image cropping over the whole dataset. This may be approached in a few different ways, two of which are following:

1. Cropping out the femurs layerwise using the exact dimensions of their respective bounding boxes is an obvious idea which arrives at the true global minimum of our cropping problem. However, the trouble of this approach is that each axial femur layer would then have different dimensions defined by its respective bounding box. This complicates the subsequent construction of patches and one would have to introduce some sort of padding in order to unify the shapes of the layers. This would ultimately hurt the continuity of the image in the patch and therefore this method is not suitable.

2. An opposite approach requires finding the smallest dimensions which are able to fit all the femurs in our dataset. Although these constants are obviously not globally optimal, after investigation we found that they are acceptably close and the benefit of simplicity overweights their suboptimality. A noteworthy fact is that we examined this approach on the femurs which had their joints already removed. This is important because joints would have a serious impact on the result of this method because of their shapes, as explained earlier in this chapter. Martínez et al. arrived to a similar solution to this problem in [68].

After a thorough discussion and a study of the dataset statistics, we agreed to use the second mentioned method and proceeded to determine these constants to both have a value of 63 pixels. To reiterate, these values represent the smallest width and height of a bounding cuboid that is capable of encasing all femurs in our dataset so that the joints are not included. The fact that the constants are equal is a convenient coincidence.

3.3.2 Augmentations

Since our dataset is annotated only on the femur level we first had to deal with the absence of labels for individual patches. Without a manual annotation there was no simple way of extracting infiltrated volumes automatically from the images. For that reason we reversed the process and create artificial lesions by augmenting images of healthy femurs. We employ this technique to be able to assess the classifier performance under the weakly supervised learning paradigm while observing and evaluating its predictions on the level of individual instances. Further details are provided in chapter 5.

We use two different methods to imitate the infiltration of multiple myeloma:

1. Creating a small 3D ellipsoid randomly positioned in the bone marrow volume of a femur CT image to simulate a homogeneous lytic lesion.

2. Adding a random noise to a marrow volume representing a diffuse lesion.

For training purposes, these two augmentations were applied on healthy patches which were then labelled as positive. The application of either of them is determined by their respective probabilities as adjustable hyperparameters. Following subsections briefly describe both augmentations in general terms. There are many Implementation details which are quite technical, and their description is therefore omitted in the text. A detailed explanation of the whole process is provided in the code documentation. All possible appearances of an augmented patch in are shown on figures 3.9 and 3.10 from axial and sagittal view.

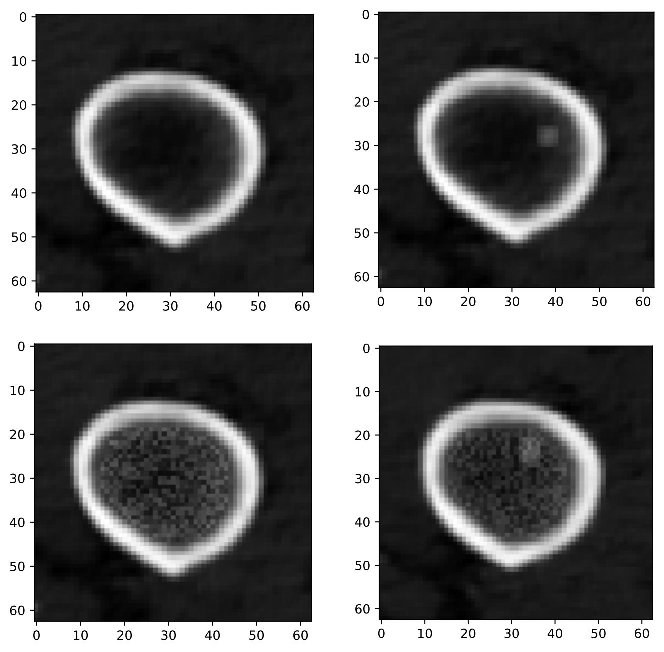


Figure 3.9: Axial view of a healthy femur marrow (top left), with an artificial ellipsoid lesion (top right), with a diffuse noise lesion (bottom left) and with both augmentations (bottom right). These lesions were generated with a *real* level of intensity.

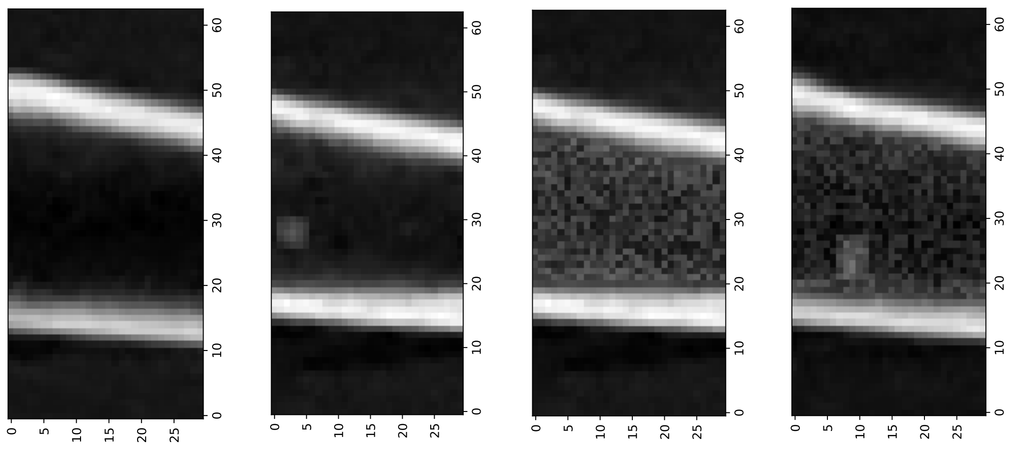


Figure 3.10: Sagittal view of the four augmentations from figure 3.9.

Noise lesion

Since our dataset contains segmented binary marrow masks for all cropped femur CT images, adding a noise to the marrow may be done by simply generating a random noise, adding it to the marrow mask, multiplying by a desired intensity factor and adding the result to the original image. The factor determines the intensity of the noise in proportion to the rest of the image.

Ellipsoid lesion

In order to generate an authentic lesion, one must deal with a number of technical difficulties. Most importantly, it must be assured that the whole lesion body stays in bounds of the marrow volume. This problem requires a few programmatic checks which may increase the overall computational cost of the training considering that these operations take place in a three-dimensional space and they are being executed with every sample of every training mini-batch. According to our measurements, it is not a limiting factor of the whole computation process though.

The mentioned intensity factors belong to one of three categories – *mild*, *real* or *hard* depending on their intensity relative to the patch intensities normalized to range from 0 to 1. To promote the randomness of the data, the three categories represent short intervals of values. During each individual augmentation a factor is sampled from an interval of the user-defined category. Table 3.2 summarizes the intervals for each interval and for both augmentation types.

Note that the intensity intervals differ for both lesion augmentations. This is because the *real* interval was purposefully manually adjusted so that the final lesions are appear authentic. The *mild* and *hard* categories are meant for experimental testing of the factor influence on the classifier performance.

Lesion type	Intensity level		
	Mild	*Real*	*Hard*
Ellipsoid	$[0.08, 0.17]$	$[0.18, 0.24]$	$[0.25, 0.35]$
Noise	$[0.05, 0.09]$	$[0.1, 0.19]$	$[0.2, 0.3]$

Table 3.2: The augmentation intensity levels and their respective intervals.

3.3.3 Transformations

The last step in our data flow pipeline is the application of non-destructive transformations on patches. Namely, we employ two simple operations, a rotation and a flip. Similarly to the lesion augmentations, these transformations are also tied to an adjustable probability of effect. In our implementation, the degree of both of these transformations is also randomized. The purpose of this step is to artificially expand the dataset and improve robustness of our network. This technique is usually called data augmentation in deep learning practice but in the context of this thesis this term is reserved for the operations of creating the artificial lesions. Rotation, flipping, scaling, shifting, cropping etc. are common augmentation used in neural network training to simulate a broader space of inputs which reportedly leads to a better generalization of the final model.

These transformations are naturally additive, which means that in total one sample has twelve different orientations (3 flip $\times$ 4 rotational).

4 | Implementation

Contents

4.1 Software design **59**
4.1.1 Tools 59
4.1.2 Data-manipulation hierachy 60
4.2 Classifier design **60**
4.2.1 Base network 61
4.2.2 MIL modifications 61
4.3 Training settings **64**

This chapter is dedicated to description of the prototype software implementation and tools chosen for its development. It also reviews our classifier design and general training settings and discusses several issues and obstacles which were solved in the process. The whole repository is stored online and can be retrieved from `https://gitlab.fel.cvut.cz/machvojt/dp`.

4.1 Software design

The whole project was designed with attention to modularity of the individual parts. Since the final repository is quite extensive, only a few selected parts are described in this section.

4.1.1 Tools

The entire project including all scripts was written in Python programming language with the help of common high-performance computational libraries like NumPy and SciPy for efficient data handling. PyTorch deep learning framework was used for building, training and evaluation of the classifier model. PyTorch is a modern open source high-level library for fast and convenient implementation of neural networks with APIs for Python and C++. It is popular for its intuitive syntax, efficient backend computation structures and usage of dynamic computation graphs, which are more flexible and user-friendly than static graphs in the case of TensorFlow.

PyTorch also supports a connection with NVIDIA CUDA technology for efficient model training on dedicated graphics hardware. All resource-expensive training sessions were run remotely on faculty servers *zorn* and *boruvka*, which host arrays of eight NVIDIA GTX 1080 Ti GPUs.

4.1.2 Data-manipulation hierachy

A large portion of the code in the software project deals with data loading and data processing, which is implemented by several classes arranged in a hierarchical structure with separated responsibilities. Following list describes these classes from the lowest level to the highest level and figure 4.1 provides its schematic illustration.

- `DS` class – A low-level class providing file handling and I/O operations. It is able to retrieve femurs or other specific files related to a given patient and vice-versa. This functionality exploits a fixed naming convention of our data.
- `PatchGenerator` class – A class managing femur loading and patch generation. Patches are supplied from currently loaded femur, if it is depleted, next femur is loaded automatically. Moreover, it keeps track of metadata of the current femur like the patient ID and or a leg indicator.
- `SVDataset` and `MILDataset` classes – Indirect descendants of the `Dataset` class from the PyTorch library in the class inheritance system. These classes perform higher-level tasks under either supervised or multiple-instance learning paradigm including a management of lesion augmentations and tensor transformations described in section 3.3. They aggregate all the related data and metadata into objects of custom classes like `Patch`, `Bag` and `Batch`, which is not necessary in the runtime but it is useful for testing.
- `SVDataloader` and `MILDataloader` classes – These classes mimic the data-batching functionality of the `torch.utils.data.Dataloader` class which could not be used in our MIL setting due to non-standard batching and custom training loop as explained in 5.4.

4.2 Classifier design

As a classification model, we designed a CNN with 3D convolutions. An alternative choice of 2D convolutions with a channel dimension substituting for the third data dimension was also considered, although after discussion we agreed that 3D architecture is more suitable for our data of 3D CT images. We faced many design-related decisions since the employment of a CNN was a novel approach to lesion detection in femurs.

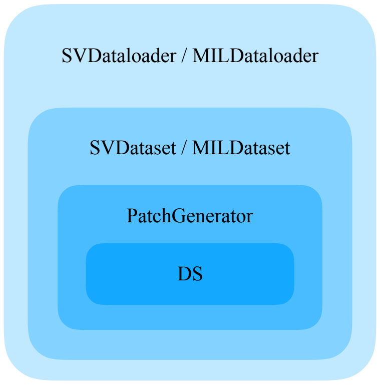


Figure 4.1: A nested structure of data loading classes used in the implementation of our thesis project. The innermost is the lowest-level class.

4.2.1 Base network

An inspiration for the network architecture has been drawn from some of the research papers mentioned in 2. Since the lesions in our task are quite symmetrical objects in gray scale, and the network output is only a probability measure, we assumed that a simpler network would have a sufficient capacity to perform well, given appropriate training conditions. Therefore, we decided to use a slightly modified architecture from [76], perform tests and switch to a more complex model if necessary. The network was then appended with a sigmoid output layer which outputs a single probability measure. Table 4.1 presents the chosen architecture.

This network, hereinafter referred to as the *supervised network* or basenet, was used in the supervised experiments and in the transfer learning setting described in the next chapter.

4.2.2 MIL modifications

As described in 1.2.6, standard multiple instance learning assumes bag-level labels only for the data. In our case, a single femur represents a bag, femur patches are its instances and the bag label is the binarized diagnosis associated with the femur.

Aggregation layer

We modified our supervised network to be compatible with the MIL setting by appending it with an aggregation layer. Figure 4.2 shows a diagram of the modification. In the rest of the text, this network will be referred to as the *mil network*.

For the aggregation layer we employed Noisy-And function proposed by Kraus et al. in [82]. It is a parametric sigmoid function that maps the instance label

Layer	Number of kernels	Kernel size
C	32	3x3x3
BN	32	–
R+M	–	2x2x2
C	32	3x3x3
BN	32	–
R+M	–	2x2x2
C	16	3x3x3
BN	16	–
R+M	–	2x2x2
F	128 units	
F	64 units	

Table 4.1: The proposed architecture of our CNN model. C, M stand for 3D variants of convolutional layer and max-pooling layer respectively. BN and F are batch normalization and fully connected layers, R stands for ReLU activation layer.

predictions to the bag label prediction by the following general formula:

$$P_i = \frac{\sigma(a(p_{i\bar{j}} - b_i)) - \sigma(-ab_i)}{\sigma(a(1 - b_i)) - \sigma(-ab_i)}, \tag{4.1}$$

where

$$p_{i\bar{j}} = \frac{\sum_j p_{ij}}{|j|} \tag{4.2}$$

denotes the mean of instance level predictions p_j of class i.
Furthermore, a is defined as a global non-trainable hyperparameter and b_i is a trainable parameter of class i, all of which determine the shape of the Noisy-And sigmoid curve. Note that in our case of a binary prediction task, an output of the classifier is merely a single probability of the input being positive, hence we can omit the class index i from the equations.

Custom input batching

Prior to the learning stage, we had to overcome an issue regarding a batching process of input tensors supplied to the classifier. Our supervised network classifier was designed to batches of uniformly-shaped patches. Specifically, we unified patch dimensions to values of 63 pixels for width and depth and 30 pixels for height, although these values are externally adjustable. Each patch is represented as a four dimensional tensor with the channel dimension equal to one

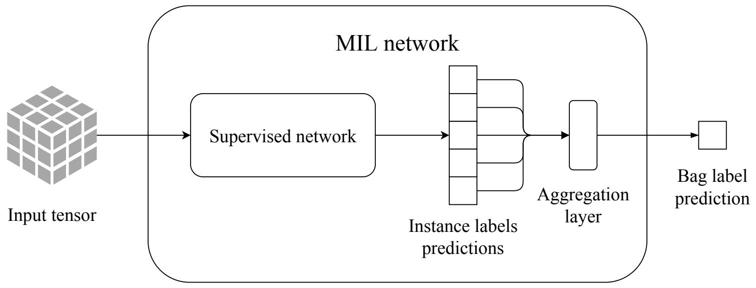


Figure 4.2: A schematic diagram of the supervised network and its role in the mil network.

since CT images have one color channel. Patches are drawn from the dataset in a controlled fashion regardless of their femur affiliation in order to populate the batch given a required batch size. This means that the number of patches in a femur is not important for the final batch size. Should there be a shortage of patches, i.e. when the generating dataset runs out, the last batch does not need to have the full capacity. This is not a problem because the batch dimension does not affect the computation and thus it is allowed to vary.

A single forward pass of an input batch of N patches through the supervised network produces N label predictions. In tensor notation this operation might be described as

$$N \times C \times D \times H \times W \rightarrow N \times 1,$$

where N is the batch dimension. Note that the input is a 5D tensor.

This is a standard batching process in a fully supervised setting, where individual input tensors have a unified shape. In our MIL setting though, where each femur should be treated as a single data point, this is does not hold since femurs have variable lengths. The input to the mil network is a single femur divided into K patches, where K depends on the femur length. Batching multiple femur tensors is therefore impossible unless they are reshaped to a fixed length. Moreover, note that even if we unified the values of K across the whole dataset, a standard batching would introduce an additional dimension resulting in a 6D input tensor.

We solved this problem by matching the 5D input tensor with the 5D femur tensor substituting the batch dimension with the K femur dimension to obtain

$$K \times C \times D \times H \times W \rightarrow K \times 1.$$

This allowed us to keep the base network architecture unchanged for MIL setting, but we also had to substitute for the missing batching process in order to maintain the stochastic gradient descent training properties. A standard training performing a minibatch SGD update is described by a PyTorch-like pseudocode in listing 4.1.

```
# draw a batch of patch-label pairs from the dataset
for batch, labels in data_loader:
    # clear all previously stored gradients
    optimiser.zero_grad()
    # N x 1 tensor of label predictions
    preds = model(batch)
    # backward done for the whole batch at once
    l = loss.backward()
    # parameters update controlled by an optimiser
    optimiser.step()
```

Listing 4.1: Standard SGD algorithm.

We modified this training routine by using an accumulated gradient update and custom data loaders. Instead of creating a batched tensor object, which was not possible for the reasons explained above, we kept the individual femurs (bags) in a simple list, performed the `loss.backward()` method for each processed bag and called the `optimiser.step()` only after depleting the list. A pseudocode in listing 4.2 explains this process.

```
# batch_list is a list of (bag, label) pairs
for batch_list in data_loader:
    # clear all previously stored gradients
    optimiser.zero_grad()
    for bag, bag_label in batch_list:
        # bag label prediction
        pred = model(bag)
        # accumulation of loss gradients over individual bags
        l = loss.backward()
     # SGD step using the accumulated gradient over all bags in a batch
    optimiser.step()
```

Listing 4.2: Our SGD modification using accumulated gradient.

4.3 Training settings

In all training cases models were optimised using Adam optimizer [83] with default parameters and binary cross-entropy was used as a loss measure. To track the training progress our training script records three training losses, each averaged over different number of training steps – batch loss, epoch loss and total loss. Validation of the model is performed every n training epochs, where n is a user-defined value set to one by default. Prior to each training session the dataset is split to the conventional three disjunctive sets used for training, validation and testing of the model by a user-defined split ratio.

The training is moderated by a `ReduceLROnPlateau` scheduler, which reduces learning rate of an optimizer after a predefined number of non-improving validation results during training. This technique reportedly helps in some cases to converge to better local optima. Finally, we use early stopping to track the progress of the validation error through training and avoid overfitting the model.

5 | Experiments

Contents

5.1 Input arguments **67**
5.2 General settings **68**
5.3 Supervised learning experiment **69**
5.3.1 Training 69
5.3.2 Evaluation 70
5.4 Multiple instance learning experiment **70**
5.4.1 Standard MIL 70
5.4.2 Transfer learning 72

We carried out our experiments in two main phases. First, we worked under a standard fully supervised learning paradigm to assess the quality of our model of choice, then we proceeded to multiple instance learning setting. Both experiments were executed multiple times with several different combinations of hyperparameters. Finally the results were averaged, evaluated and compared.

5.1 Input arguments

Apart from the standard design-related hyperparameters described in section 1.2.4, we introduced a set of parameters specific for our task, which control the data augmentation process (Tab. 5.1), input masking (Tab. 5.2) and more. These parameters are defined by the user via command line arguments.

Note that under MIL setting, the *patch probability* is also conditioned on the *femur probability* and for a patch $\mathbf{p}$ in femur $\mathbf{f}$ holds

$$p(y_\mathbf{p}|\mathbf{p}, \mathbf{f}) = p(y_\mathbf{p}|\mathbf{p}) \cdot p(y_\mathbf{f}|\mathbf{f}),$$

where $y_\mathbf{p} \in \{0, 1\}$ and $y_\mathbf{f} \in \{0, 1\}$ are patch and femur labels. The augmentation process assigns the femur a label that is determined by its patch labels as

$$\max y_\mathbf{p} \text{ for } \mathbf{p} \in P_\mathbf{f},$$

where $P_\mathbf{f}$ is a set of patches belonging to a femur $\mathbf{f}$.

The user is also able to choose any combination of three input masking options presented in table 5.2. For example, combination $\{1, 2\}$ masks out cortical bone

Parameter	Value range	Meaning
Augmentation level	{mild, real, hard}	Intensity of artificial lesions relative to a normalized patch voxel intensity
Patch probability	[0,1]	Probability of generating a lesion in given a patch $\mathbf{p}$, i.e. $p(y_p = 1\|\mathbf{p})$
Femur probability	[0,1]	Probability of generating a lesion in a given femur $\mathbf{f}$, i.e. $p(y_f = 1\|\mathbf{f})$

Table 5.1: Table of parameters that control the data augmentation. First two apply for both supervised and MIL experiments; the last one is MIL-specific.

Mask value	Segmented tissue
0	Outer region (fat and muscle tissue surrounding the femur)
1	Cortical bone
2	Bone marrow

Table 5.2: Input image mask values and their respective regions of effect.

and bone marrow and removes all surrounding tissues. Masking is used in both experiments to assess the influence of image noise on the result performance.

The scripts in the attached repository provide quite a large number of user-adjustable parameters. For a detailed description refer to the code documentation.

5.2 General settings

In both experimental settings a few combinations of hyperparameters and arguments were selected and three or more training runs were performed for each combination. All three runs under each given setting used the same testing dataset.

Training and testing datasets are created by splitting the list of patients into two random disjunctive sets. The training set is then further split to training and validation set. All the splitting ratios are adjustable by command line arguments by they were left to default values in our experiments, which is 6 : 1 for train:test split ratio and 5 : 1 for train:val ratio.

The final models are evaluated by ROC curves and AUC scores. All ROC curves in this chapter were generated using the best saved model, i.e. the one with the lowest validation loss from all saved models in the given training session.

5.3 Supervised learning experiment

Supervised learning experiments were prepared to assess the capabilities of our chosen network architecture. For that purpose we examined six combinations of values of three parameters – augmentation level, patch augmentation probability and marrow masking.

5.3.1 Training

Figure 5.1 presents convergence rates of the parameter combinations each averaged over multiple runs.

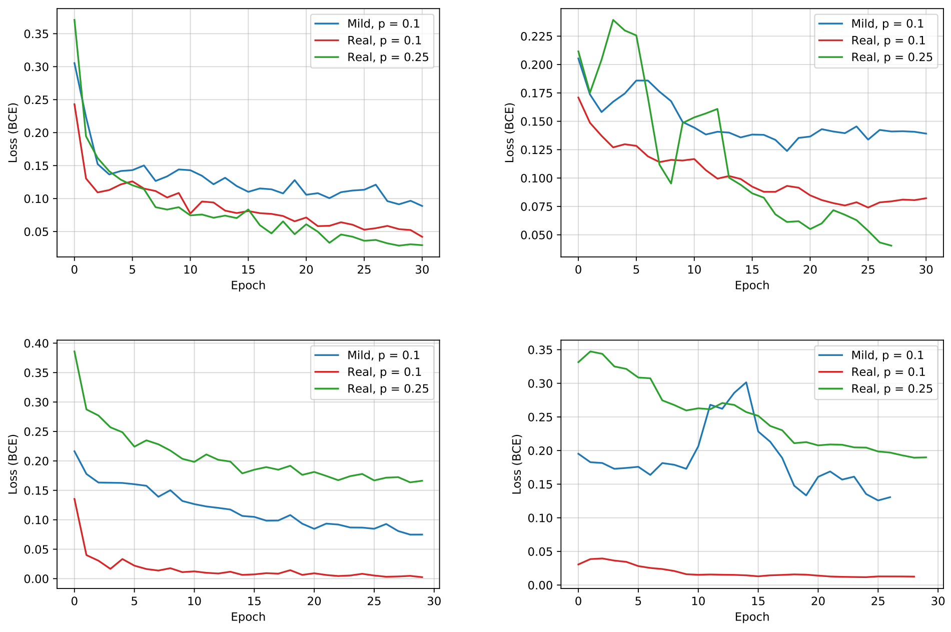


Figure 5.1: Training (left column) and validation (right column) loss on samples without any masking (top) and with marrow masking (bottom).

Note that the models were trained with the early stopping scheduler mentioned in chapter 4, which halted the training at a different epoch in each of the runs. In order to the gather the data in concise graphs, the value arrays needed to be truncated to a common number of thirty epochs here. The training sessions were actually longer with an upper limit of fifty, seventy or one hundred epochs.

5.3.2 Evaluation

Evaluations of all the experimental runs are presented on figure 5.2. The augmentation level seems to be the defining factor, marrow masking on the contrary has a negligible effect on the results. This implies that the convolutional network does not struggle with the noise information in the input images as much as was assumed.

5.4 Multiple instance learning experiment

Standard multiple instance learning (MIL) works with bag-level labels only. In our case a femur represents the bag, its instances are the femur patches and the bag label is the label associated with the whole femur. In the training phase of the MIL experiment, the patches were augmented on the run similarly as in the previous experiment, hence the instance-level labels were known but they were not available to the classifier.

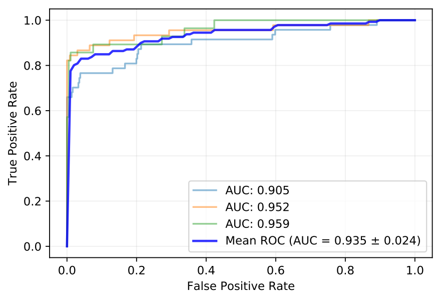
The mil network classifier described in chapter 4 is used in all experiments mentioned in this section. Since we wanted to compare the learning abilities of the classifier under both experimental scenarios, we extended the mil network to also output the intermediate instance-level predictions. Patches of each input femur are augmented on the run, therefore the ground truth instance labels are known but they are not available to the MIL classifier. This allows us to measure the learning progress on the level of instances in the MIL setting.

We examined two approaches to training the mil network – learning from scratch and by using a pre-trained supervised network from the previous experiments.

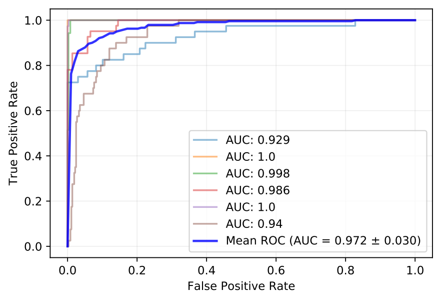
5.4.1 Standard MIL

The idea of the standard MIL is to optimise a classifier only using the loss signal measured between the predicted bag label and the ground truth bag label. We performed multiple experimental runs with different parameters in the MIL scenario. Figure 5.3 summarizes the training progress of all the runs.

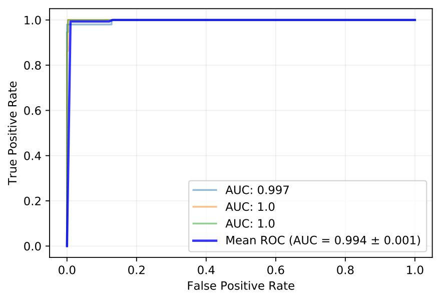
The results of this experiment did not quite meet our expectations – it is clear that the model does not converge in this setting. Upon deeper investigation of the possible cause, we found that the model fails to differentiate between predictions of individual instance labels i.e. it assigns very similar values usually close to zero or one to all instance predictions of an input batch. Another observation is that the trainable parameter b of the Noisy-And aggregation layer fluctuates throughout the whole training process. This indicates that the loss signal, which is based solely on the bag label prediction, is probably not strong enough to drive the entire network to convergence. It is also possible that the aggregation layer



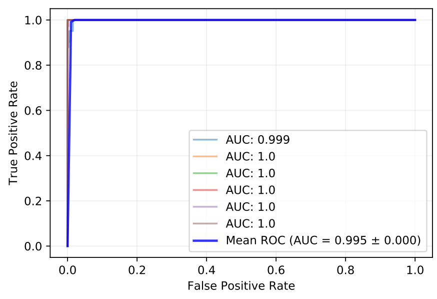
(a) Augm. level: *mild*, $p = 0.1$, no mask.



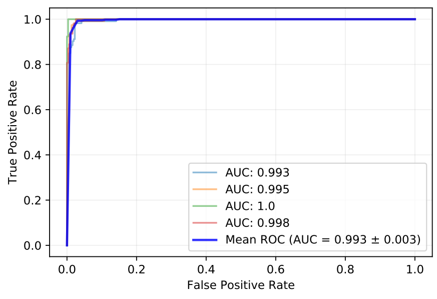
(b) Augm. level: *mild*, $p = 0.1$, marrow masking.



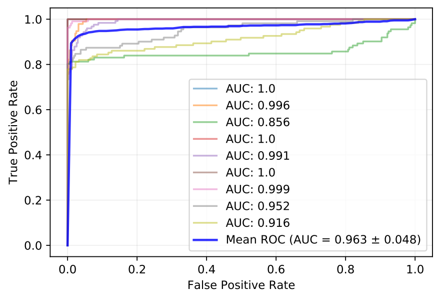
(c) Augm. level: *real*, $p = 0.1$, no mask.



(d) Augm. level: *real*, $p = 0.1$, marrow masking.



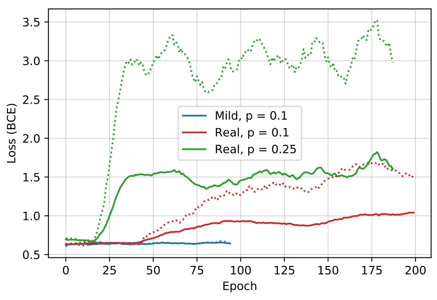
(e) Augm. level: *real*, $p = 0.25$, no mask.



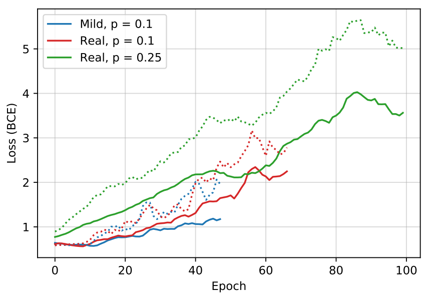
(f) Augm. level: *real*, $p = 0.25$, marrow masking.

Figure 5.2: Plots of ROC curves for fully supervised experimental runs with different parameter combinations. Right column shows runs with marrow masking of the input patches.

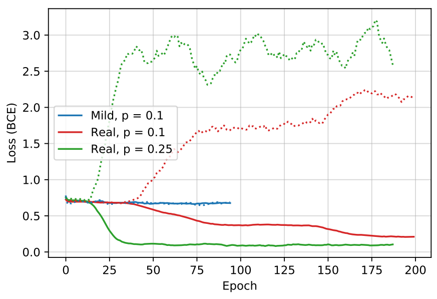
parameters interfere with the rest of the network parameters during the training. A few potential improvements regarding this issue are proposed in the conclusion of this thesis.



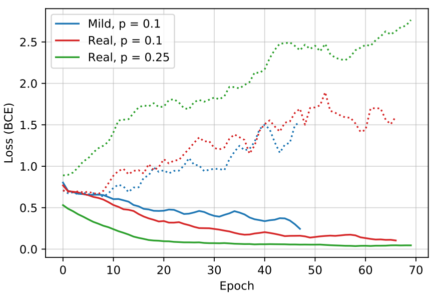
(a) Instance-level losses without masking.



(b) Instance-level losses with marrow mask.



(c) Bag-level losses without masking.



(d) Bag-level losses with marrow mask.

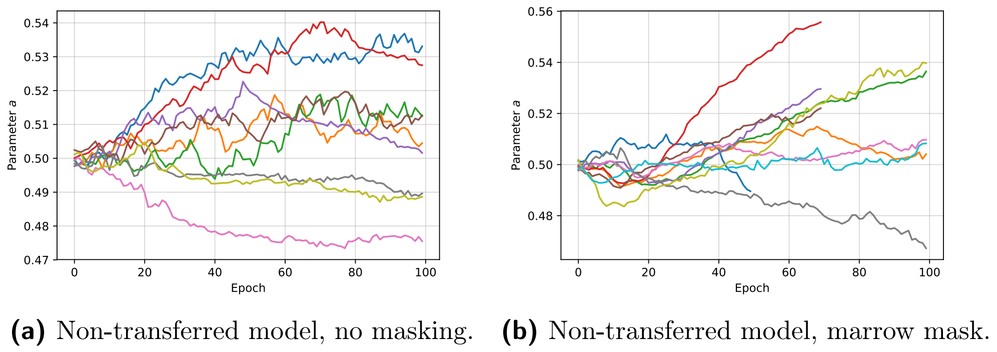
Figure 5.3: Training progress of different MIL settings. Full lines represent training loss, dotted lines validation loss. All plots were generated by averaging multiple runs. Marrow masking seems to have no positive effect on the training. Validation losses diverge in all test cases.

5.4.2 Transfer learning

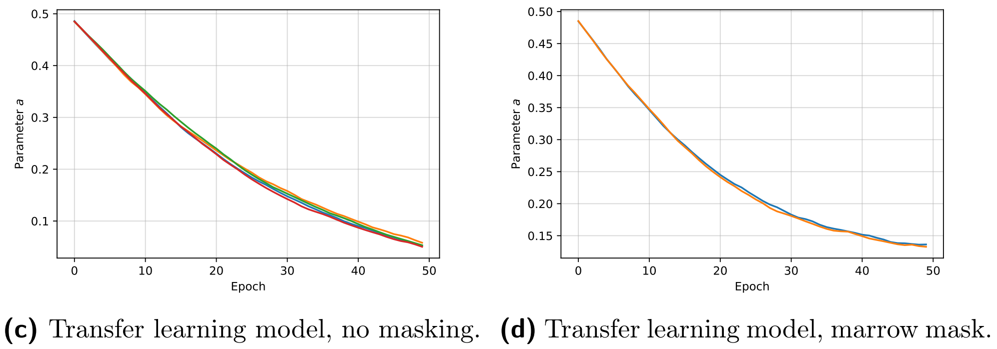
The goal of transfer learning is to reuse the knowledge extracted from one input domain on another similar domain. It is a popular method in practice particularly in terms of neural networks because it saves time and resources required to train large complex models from the start. A common approach is to take a previously trained model, freeze most of its parameters and train only a few last layers. This is especially useful in convolutional networks, which aim to embed low-level input domain features in its frontal layers, which should theoretically be universally valid across similar domains.

The design of our model allowed us to use a pre-trained model of the supervised network as a basis for the mil network and transfer the learned instance-level knowledge. The weights and biases of the base network were frozen and only the parameter b of the Noisy-And aggregation layer was trained. Training and validation losses averaged over multiple runs are presented on figure 5.5.

We tested this approach in order to observe the behaviour of the Noisy-And



(a) Non-transferred model, no masking. **(b)** Non-transferred model, marrow mask.



(c) Transfer learning model, no masking. **(d)** Transfer learning model, marrow mask.

Figure 5.4: Non-averaged values of the Noisy-And parameter $b \in [0, 1]$ during individual runs of training in the standard MIL setting (top) and using a transferred supervised network (bottom). In all cases b was initialized to 0.5. Apparently the parameter b is unable to converge if trained simultaneously with the other network parameters.

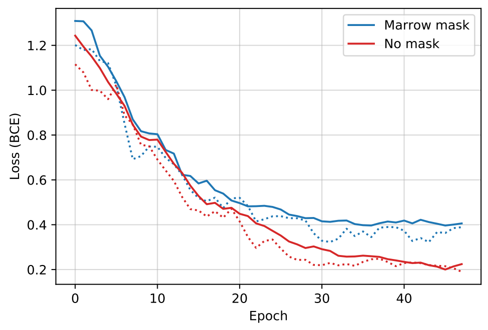


Figure 5.5: Training progress of MIL setting with pre-trained supervised network with and without marrow masking. The transferred fully supervised model was trained with *real* augmentation level and 0.25 patch prevalence. Full lines represent training loss, dotted lines validation loss.

aggregation in a situation where the instance label predictions are already mostly correct. We found that for unexplained reasons the optimizer drives the value of the parameter b below zero even though the authors in [82] state that b should be in the range of $[0, 1]$. The development of b during training with different initialization values are shown on figure 5.6.

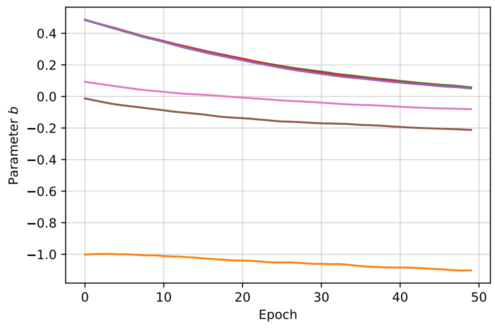


Figure 5.6: Convergence of the Noisy-And parameter b during training sessions with different initialization values.

The cause of this unexpected behaviour is not clear, although performance of the trained models seem to benefit from the negative values of b (Fig. 5.7).

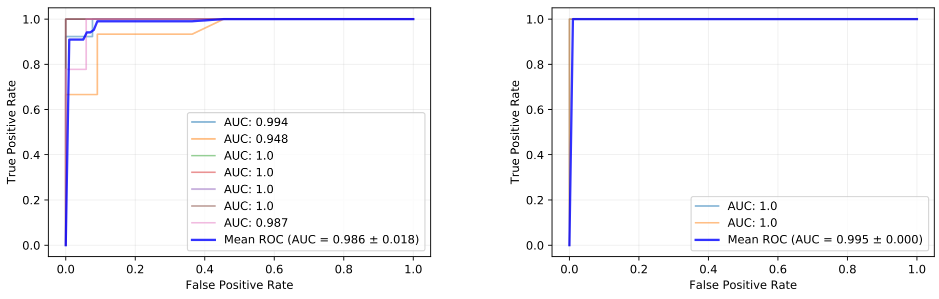


Figure 5.7: Mean ROC curves for the MIL classifier with a transferred pre-trained supervised network. The performance on inputs without masking (left) and with marrow masking (right) is quite similar.

Conclusion

The goal of this thesis was to propose, implement and evaluate a computer aided diagnosis (CAD) system based on neural networks and deep learning in order to diagnose a multiple myeloma (MM) from computer tomography images of femurs. The clinical diagnosis of the disease is commonly determined by a presence of lytic lesions observed in bone marrow of the majority of MM patients. This was reflected in the design of our CAD system prototype.

A thorough data cleaning and preprocessing was performed to normalize the available data to the best possible extent. This included detection of bone implants, femur alignment and air-voxel removal. Furthermore, a heuristic was designed to find suitable cropping range in each bone image to consistently exclude joints on both ends. Altogether, the data preprocessing resulted in a significant reduction of the input tensor size, which aids any future image recognition system learning from this dataset.

All parts including model training and evaluation as well as data manipulation were implemented within own extensive framework, which was designed with an accent on modularity and reusability for similar future projects. The complete framework uses and extends the capabilities of the PyTorch module and was made available through a gitlab repository.

We designed a binary classifier based on a 3D convolutional neural network for the purpose of detecting the bone marrow lesions in input volumes. Since our dataset was only annotated on the level of femurs without further localization of the lesions, we also explored the possibilities of multiple instance learning (MIL) to tackle the weak supervision. In order to evaluate the model learning in the MIL setting we developed complex procedures of data augmentation like artificial lesion generation to simulate a full supervision from the original task.

We experimented with different input data variants – prevalence of positive instances in the dataset, strength of data augmentation and different masking settings. Three distinct learning scenarios were examined – a fully supervised learning, a standard multiple instance learning and multiple instance learning with pre-trained instance classifier. All experiments were run multiple times and the results were averaged to eliminate the role of chance. Despite the relatively small dataset and a rather simpler network architecture we observed a solid classification performance from the fully supervised learning experiment. It was revealed that the augmentation factor has a larger impact on the performance than the patch prevalence value. This might be problematic in diagnosis of patients with less pronounced lesions as the classifier might be susceptible to increased false negative rate. The benefits of marrow masking were found to be negligible or very subtle, which is surprising given the amount of noise coming from tissues surrounding the marrow mass in the femur CT images. The results obtained from the standard MIL experiments are quite poor since the model diverged in all explored training settings. We conclude that the error signal based solely on a bag label prediction is not strong enough to drive the optimisation of the entire model in this scenario to convergence. We discovered that the multiple instance learning with pre-trained instance classifier achieves satisfactory performance in diagnosing input femurs. This finding confirms that the proposed network for the standard MIL setting could deliver competitive results as a CAD system if successfully trained.

Further work

We propose several improvements, which could potentially lead to better results of our CAD system.

First, since deep learning is a resource-expensive discipline, we were only able to test a few selected combinations of hyperparameters. An extensive hyperparameter optimisation was infeasible for time reasons. It is plausible, that a different choice of values would yield better results. The effect of different training schedulers and optimizers could be examined as well as the role of different, possibly non-trainable, aggregation layers.

Secondly, the poor performance observed in the multiple instance learning setting might presumably be improved by incorporating instance-level network predictions to the loss function in certain cases. For illustration, working under the standard MIL assumption, when a healthy femur is passed to the classifier all its patches can be labelled as negative. This information could be exploited in order to compute an additional loss using instance-level predictions along with the standard bag-level prediction loss. The design of the proposed network allows this without larger modifications.

Lastly, a certain kind of heuristic approximating positions and spread of the marrow lesions in a given femur could be employed to create new annotated training data for the network classifier. Some of the probabilistic models from the related work in this area could be possibly explored for this purpose.

Bibliography

[1] Z. V. Maizlin and P. M. Vos, “Do we really need to thank the beatles for the financing of the development of the computed tomography scanner?,” *Journal of Computer Assisted Tomography*, vol. 36, no. 2, pp. 161–164, 2012.

[2] M. J. Willemink and P. B. Noël, “The evolution of image reconstruction for CT—from filtered back projection to artificial intelligence,” *European Radiology*, vol. 29, pp. 2185–2195, Oct. 2018.

[3] M. M. Lell *et al.*, “Evolution in computed tomography,” *Investigative Radiology*, vol. 50, pp. 629–644, Sept. 2015.

[4] G. Liugang *et al.*, “Effects of 16-bit CT imaging scanning conditions for metal implants on radiotherapy dose distribution,” *Oncology Letters*, Dec. 2017.

[5] E. Kreit *et al.*, “Biological versus electronic adaptive coloration: how can one inform the other?,” *Journal of The Royal Society Interface*, vol. 10, Jan. 2013.

[6] J. C. Mandell *et al.*, “Clinical applications of a CT window blending algorithm: RADIO (relative attenuation-dependent image overlay),” *Journal of Digital Imaging*, vol. 30, pp. 358–368, Jan. 2017.

[7] *Computed Tomography.* Springer Berlin Heidelberg, 2008.

[8] B. G. Hansford and R. Silbermann, “Advanced imaging of multiple myeloma bone disease,” *Frontiers in Endocrinology*, vol. 9, Aug. 2018.

[9] C. Röllig, S. Knop, and M. Bornhäuser, “Multiple myeloma,” *The Lancet*, vol. 385, pp. 2197–2208, May 2015.

[10] C. Gerecke *et al.*, “The diagnosis and treatment of multiple myeloma,” *Deutsches Aerzteblatt Online*, July 2016.

[11] “Multiple Myeloma.” `https://www.cancerquest.org/patients/cancer-type/multiple-myeloma`. [Online. Accessed 20-July-2020].

[12] “Case courtesy of Assoc. Prof. Frank Gaillard, Radiopaedia.org, rID: 16464.” `https://radiopaedia.org/articles/endosteal-scalloping`. [Online. Accessed 10-July-2020].

[13] "Case courtesy of Dr Matthew Lukies, Radiopaedia.org, rID: 55119." `https://radiopaedia.org/articles/multiple-myeloma-1`. [Online. Accessed 10-July-2020].

[14] D. Resnick, *Diagnosis of bone and joint disorders.* Philadelphia: Saunders, 2002.

[15] R. Eslick and D. Talaulikar, "Multiple myeloma: from diagnosis to treatment," *Australian Family Physician*, vol. 42, pp. 684–688, Oct. 2013.

[16] S. Jewell *et al.*, "Multiple Myeloma: Updates on Diagnosis and Management," *Federal practitioner: for the health care professionals of the VA, DoD, and PHS*, vol. 32, pp. 49S–56S, Aug. 2015.

[17] R. Zambello *et al.*, "Whole-body low-dose CT recognizes two distinct patterns of lytic lesions in multiple myeloma patients with different disease metabolism at PET/MRI," *Annals of Hematology*, vol. 98, pp. 679–689, Dec. 2018.

[18] "Implementing deep learning using cuDnn." `https://www.slideshare.net/deview/251-implementing-deep-learning-using-cu-dnn/4`, 2015. [Online. Accessed 24-July-2020].

[19] M. Alkhayrat *et al.*, "A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA," *Journal of Big Data*, vol. 7, Feb. 2020.

[20] S. Russell, *Artificial intelligence : a modern approach.* Hoboken: Pearson, 2021.

[21] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," 2019.

[22] "Deep learning and the future of AI." `https://indico.cern.ch/event/510372/`, 2015. [Online. Accessed 24-July-2020].

[23] R. Vaillant, C. Monrocq, and Y. Le Cun, "Original approach for the localisation of objects in images," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 141, no. 4, pp. 245–250, 1994.

[24] Y. LeCun, Fu Jie Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–104 Vol.2, 2004.

[25] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," 2013.

[26] Y. LeCun *et al.*, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[28] N. Buduma, *Fundamentals of deep learning : designing next-generation machine intelligence algorithms.* Sebastopol, CA: O’Reilly Media, 2017.

[29] “CS231n Convolutional Neural Networks for Visual Recognition.” https://cs231n.github.io/convolutional-networks/#convert. [Online. Accessed 14-July-2020].

[30] M. Yani *et al.*, “Application of transfer learning using convolutional neural network method for early detection of terry’s nail,” *Journal of Physics: Conference Series*, vol. 1201, p. 012052, May 2019.

[31] Y. Xu *et al.*, “Scale-invariant convolutional neural networks,” 2014.

[32] M. Jaderberg *et al.*, “Spatial transformer networks,” 2015.

[33] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” 2013.

[34] “Learn About Convolutional Neural Networks.” https://www.mathworks.com/help/deeplearning/ug/introduction-to-convolutional-neural-networks.html. [Online. Accessed 24-July-2020].

[35] S. Srivastava *et al.*, “Handwritten digit classification using Convolutional Neural Networks,” *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 6, pp. 301–305, 5 2020.

[36] J. Deng *et al.*, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[38] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” 2015.

[40] J. Hu *et al.*, "Squeeze-and-excitation networks," 2017.

[41] J. Kim, J. Hong, and H. Park, "Prospects of deep learning for medical imaging," *Precision and Future Medicine*, vol. 2, pp. 37–52, June 2018.

[42] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013.

[43] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[44] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, Oct. 2017.

[45] T. B. Brown *et al.*, "Adversarial patch," 2017.

[46] Z. Wu, S.-N. Lim, L. Davis, and T. Goldstein, "Making an invisibility cloak: Real world adversarial attacks on object detectors," 2019.

[47] D. Hendrycks *et al.*, "Natural adversarial examples," 2019.

[48] "About Face ID advanced technology." `https://support.apple.com/en-us/HT208108`.

[49] "Tesla Autopilot AI." `https://www.tesla.com/cs_CZ/autopilotAI`.

[50] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," Master's thesis, Josef Hochreiter Institut fur Informatik Technische Universit at Munchen, 6 1991.

[51] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, 01 2010.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[55] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[56] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016.

[57] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” 2016.

[58] B. Zoph *et al.*, “Learning transferable architectures for scalable image recognition,” 2017.

[59] E. Real *et al.*, “Large-scale evolution of image classifiers,” 2017.

[60] R. Miikkulainen *et al.*, “Evolving deep neural networks,” 2017.

[61] Y. Chen *et al.*, “Reinforced evolutionary neural architecture search,” 2018.

[62] S. Xie *et al.*, “Snas: Stochastic neural architecture search,” 2018.

[63] T. Dietterich *et al.*, “Solving the multiple instance problem with axis-parallel rectangles,” *Artificial Intelligence*, vol. 89, pp. 31–71, 03 2001.

[64] Z.-H. Zhou *et al.*, “Multi-instance multi-label learning,” *Artificial Intelligence*, vol. 176, pp. 2291–2320, Jan. 2012.

[65] T. Gärtner *et al.*, “Multiple-instance kernels,” *Proc. Int’l Conf. Machine Learning (ICML)*, pp. 179–186, 01 2002.

[66] N. Weidmann *et al.*, “A two-level learning method for generalized multi-instance problems,” vol. 2837, 07 2003.

[67] F. Martínez-Martínez, J. Kybic, and L. Lambert, “Automatic detection of bone marrow infiltration by multiple myeloma detection in low-dose ct,” pp. 4813–4817, 2015.

[68] F. Martínez-Martínez, J. Kybic, L. Lambert, and Z. Mecková, “Fully automated classification of bone marrow infiltration in low-dose CT of patients with multiple myeloma based on probabilistic density model and supervised learning,” *Computers in Biology and Medicine*, vol. 71, pp. 57–66, Apr. 2016.

[69] J. Hering, J. Kybic, and L. Lambert, “Detecting multiple myeloma via generalized multiple-instance learning,” in *Medical Imaging 2018: Image Processing* (E. D. Angelini and B. A. Landman, eds.), SPIE, Mar. 2018.

[70] Z. Omiotek, O. Stepanchenko, W. Wójcik, W. Legieć, and M. Szatkowska, “The use of the hellwig’s method for feature selection in the detection of myeloma bone destruction based on radiographic images,” *Biocybernetics and Biomedical Engineering*, vol. 39, pp. 328–338, Apr. 2019.

[71] P. Kowalik, *On an implementation of the method of capacity of information bearers (the Hellwig method) in spreadsheets*, pp. 31–40. 09 2014.

[72] G. Litjens *et al.*, “A survey on deep learning in medical image analysis,” *Medical Image Analysis*, vol. 42, p. 60–88, Dec 2017.

[73] B. Wu, Z. Zhou, J. Wang, and Y. Wang, “Joint learning for pulmonary nodule segmentation, attributes and malignancy prediction,” 2018.

[74] A. Bhandary *et al.*, “Deep-learning framework to detect lung abnormality – a study with chest x-ray and lung CT scan images,” *Pattern Recognition Letters*, vol. 129, pp. 271–278, Jan. 2020.

[75] Y. Li, Y. Tian, and B. Ge, “Lung cancer classification using 3d-cnn with a scheduled learning strategy,” 01 2018.

[76] X. Huang, J. Shan, and V. Vaidya, “Lung nodule detection in ct using 3d convolutional neural networks,” pp. 379–383, April 2017.

[77] W. Zhu, C. Liu, W. Fan, and X. Xie, “Deeplung: 3d deep convolutional nets for automated pulmonary nodule detection and classification,” 2017.

[78] A. A. A. Setio *et al.*, “Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge,” *Medical Image Analysis*, vol. 42, pp. 1–13, Dec. 2017.

[79] G. Armato *et al.*, “The lung image database consortium (LIDC) and image database resource initiative (IDRI): A completed reference database of lung nodules on CT scans,” *Medical Physics*, vol. 38, pp. 915–931, Jan. 2011.

[80] Q. Dou *et al.*, “Multilevel contextual 3-d cnns for false positive reduction in pulmonary nodule detection,” *IEEE Transactions on Biomedical Engineering*, vol. 64, pp. 1558–1567, July 2017.

[81] Y. Wang *et al.*, “Ct-image super resolution using 3d convolutional neural network,” 2018.

[82] O. Z. Kraus, L. J. Ba, and B. Frey, “Classifying and segmenting microscopy images using convolutional multiple instance learning,” 2015.

[83] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

Appendix A - Additional evaluations

Figure 5.8 presents plots averaged ROC curves for all twelve combinations of augmentation level from *mild* to *hard* and patch prevalence from $\{0.1, 0.2, 0.35, 0.5\}$ in the fully supervised experiment without input masking. Both parameters have a visible impact on the classification performance, the augmentation level seems to be a slightly dominant factor.

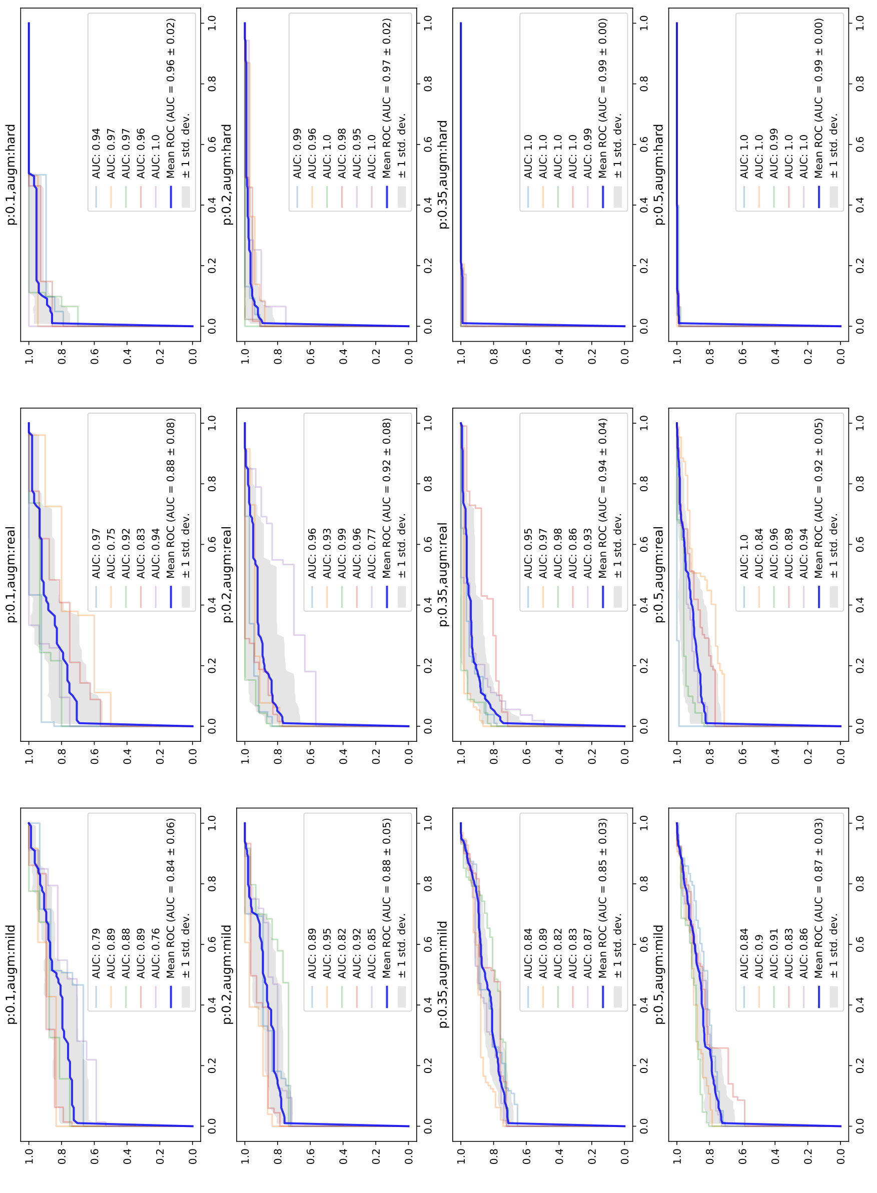


Figure 5.8: Plots of ROC curves showing the impact of selected values of augmentation levels and patch prevalence.

Appendix B - Contents of the project repository

```
dp
├── resources
│   ├── trim_dict_all
│   └── implants_all.npy
├── exec
│   ├── train_sv.py
│   ├── train_mil.py
│   ├── eval_sv.py
│   └── eval_mil.py
├── src
│   ├── augmentations.py
│   ├── constants.py
│   ├── metrics.py
│   ├── model.py
│   ├── myexceptions.py
│   ├── dataset.py
│   ├── printer.py
│   ├── result.py
│   ├── saveutils.py
│   ├── settings.py
│   └── transformations.py
├── juputils
│   ├── jds.py
│   ├── jutils.py
│   └── filenames.py
└── oth
    ├── build_dataset.py
    └── unbuild_dataset.py
```