POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

ADDRESSING COLLABORATIVE MACHINE LEARNING CHALLENGES IN MEDICAL IMAGING

Doctoral Dissertation of:
Edoardo Giacomello

Supervisor:
Prof. Daniele Loiacono

Co-Supervisor:
Prof. Luca Mainardi

Tutor:
Prof. Letizia Tanca

The Chair of the Doctoral Program:
Prof. Luigi Piroddi

Year 3 – Cycle XXXIV

In loving memory of my father.

Acknowledgements.

First and foremost, I would like to thank my advisors, Prof. Daniele Loiacono, for the continuous advice, support, and helpfulness shown during my Ph.D. and Prof. Luca Mainardi for his inspiring insights. I would like to thank Prof. Francesco Amigoni for the invaluable advice on agent modeling and ensembling approaches, and Prof. Pier Luca Lanzi for the broad support during these years. A particular mention goes to all the people I met at Humanitas for the priceless insights on medical imaging and precious advices. I'd like to also thank all my co-authors and thesis students, which provided valuable contributions to my studies and allowed me to learn a lot from their insights. Moreover, my gratitude goes to Marco Bologna for helping me move the first steps in medical imaging and all the colleagues at the B3-Lab and DEIB that I met during my Ph.D. Finally, I want to express my most sincere gratitude to my family, particularly my mother and my partner, for their loving and never-ending support.

Abstract

Machine Learning and Deep Learning tools in Medical Imaging are promising approaches to aid physicians and radiologists in performing diagnoses. Machine Learning models that work with imaging data require massive amounts of data. Although many institutes are collaborating to produce publicly available datasets of medical images, the process of data acquisition is severely limited by different challenges. These challenges are mainly related to privacy regulations and the effort of domain experts to assess imaging data quality and produce high-quality ground truth. In turn, the difficulty of managing large datasets of medical imaging translates in a scarcity of data available for research. This Ph.D. thesis studies collaborative machine learning as a methodological approach to overcome the problem of data availability. Collaborative Machine Learning is a vast area of research that includes a set of techniques, such as Distributed Learning and Esembling Methods, to enable multi-centric studies using multiple private datasets. The main idea behind collaborative machine learning is to share knowledge instead of data to overcome potential privacy issues in exchanging sensitive data. However, this approach poses challenges that include data heterogeneity due to the population included in the datasets, and data incompleteness, due to different data acquisition standards and practices among different institutions. This work provides a general taxonomy for classifying the various approaches proposed in the literature. We analyze well-established techniques such as ensemble learning and transfer learning in the context of collaborative machine learning. Moreover, we analyze more recent contributions based on distributed learning, comparing their

performances according to data heterogeneity and privacy constraints. Our experiments study multiple approaches that exploit ensemble methods, distributed learning, and transfer learning to overcome different challenges, such as data heterogeneity, model heterogeneity, and label heterogeneity using public and private datasets. Finally, we propose our approach to image segmentation based on adversarial networks and generative adversarial networks to study possible approaches to the problem of incomplete medical imaging datasets. The results are promising, showing that collaborative learning can successfully overcome the issues above. In particular, ensemble learning methods can build a single model from multiple models with different architectures when trained on different data subsets. Moreover, distributed learning approaches proved to be a good design choice when privacy has to be attained, especially in a context of data heterogeneity. Transfer learning and embedding techniques can enable the training of custom models on smaller private datasets by exploiting the powerful feature extraction modules of Convolutional Neural Networks. Lastly, our approach based on adversarial networks proved to be promising to enable the use of multi-input segmentation models when some of them are missing, thanks to image translation.

Summary

In recent years, Deep Learning achieved impressive results in different medical diagnostic tasks, often reaching human-grade performances. However, Deep Learning requires large amounts of data. To develop machine learning models that can be effectively used in practice, researchers must face the significant problem of data availability. While collecting large image datasets to develop deep learning solutions for generic imaging tasks is relatively straightforward, in the medical field, the problem of *privacy* poses significant limitations to this approach. While the most adopted solution is to apply anonymization techniques to patient data, it is often costly in terms of time and resources and not applicable in every context. In addition, due to the complexity and amount of data, significant effort is required to generate and validate the ground truth. One possible solution to this problem is a multi-centric approach that would allow sharing the effort among different entities – e.g., hospitals – without the need to build centralized datasets.

We propose a collaborative machine learning approach to the problem, which exploits distributed learning and other machine learning techniques to exchange *information* instead of *data*, allowing different entities to exploit the knowledge acquired from the other collaborating parties. In such a setting, new issues arise. These issues include the problem of different distributions in medical data, related to differences in population, and problems related to data acquisition processes, such as the problem of missing data in multi-modality imaging datasets.

Our study investigates these issues from a methodological point of view. In particular, we first propose a general framework to classify the different

distributed learning approaches used in the literature and investigate how other popular techniques such as transfer learning can be framed in a collaborative machine learning setting. Our experiments show how to exploit different machine learning techniques, such as *ensembling methods*, *distributed learning algorithms*, and *transfer learning* to enable collaborative learning in settings where heterogeneity is present in data, in the architecture of the models, or the target labels. Moreover, we provide an approach based on adversarial networks to study the problem of segmentation and datasets with missing modalities. We used two medical imaging tasks as case-study: the Brain Tumor Segmentation and the Automated Chest X-Ray Diagnosis.

Our results are promising, showing that collaborative learning could be a feasible approach to overcome the issues above. In particular, ensembling methods can be used to design a system in a setting in which multiple deep learning models are available. Our proposed methods based on *entropy* showed to be particularly effective for classification tasks when none of the available models outperforms the others on every label. When a collaborative system is to be designed and no deep learning models are already available, an approach based on distributed learning can be optimal. To investigate this approach, we compare two recently introduced techniques – Federated Learning and Split Learning –in the context of data heterogeneity. Our results show that distributed learning can reach performances close to a centralized model while providing good results as data heterogeneity and privacy requirements are introduced. Moreover, we investigate transfer learning as a collaborative tool. First, we exploit an *embedding* technique to show how to build different machine learning models based on trees by taking advantage of the feature extraction step of several Convolutional Neural Networks. Then, we apply transfer learning to use the embeddings to train models specific to a smaller private dataset of Chest X-Rays, containing a different set of labels from the original dataset. Our results show that this approach effectively enables the training of machine learning models for imaging in a setting of data scarcity and low computational resources. Lastly, we investigate Adversarial Networks first to perform transfer learning between segmentation models trained on different modalities and then generate the modality that is eventually missing from a medical dataset. Our results also show the effectiveness of transfer learning in the context of adversarial networks, although the setting is more complex than with standard neural networks. Moreover, we show that a generative approach can allow the use of machine learning models that require multiple input modalities, with only a slight loss in segmentation performances.

Contents

CHAPTER 1

Introduction

In recent years, Deep Learning reached physician-level grade performances in different medical diagnostic tasks [1, 2]. On the other hand, Computer-Aided Diagnosis (CAD) is one of the most researched subjects for medical imaging and diagnostic radiology [3], and represents a valuable tool to help physicians in automating and accelerating time-consuming processes. While most Deep Learning approaches strive to reach or surpass human performances in object-detection tasks, they are generally considered as *black-box* models. CAD systems aim instead to offer the physicians a "second opinion" to perform their diagnosis in order to decrease the false negatives of the user. Such vision could be a successful approach to bring Deep Learning tools to medical practice, and it provides one of the primary motivations for the research of machine learning and deep learning that are specific to medical imaging [4].

Another, often overlooked, reason for the research of AI in medicine is related to the growth in the world population. From 1950 to 2015, the global population increased from 2.5 to 7.3 billion, and up to 19.3 billion people are expected in 2100. [1] This trend is strongly related to an expected

[1] `https://www.eea.europa.eu/data-and-maps/indicators/total-population-outlook-from-unstat-3/assessment-1` (access date 30/04/2021).

increase in the request of medical doctors, in particular imaging experts. The medical community already issued some warnings on the subject [2], suggesting that AI might partially fill this gap [5].

However, the research of Deep Learning tools for Medical Imaging has to face one major issue. While large amounts of data are collected daily in hospitals, achieving adequate data quality for machine learning research is still challenging and often unfeasible [6]. In our study, we investigate *collaborative machine learning* as a possible approach to overcome this issue. Collaborative Machine Learning is a field that studies the exchange of knowledge between different parties – e.g., hospitals — that have to solve a learning task and dispose of a limited amount of private data. Early approaches to develop personalized healthcare services with the use of big data has posed severe privacy concerns in the past [7], highlighting the need for particular attention in data management and privacy in the field of healthcare. Other central issues in designing a distributed learning system are related to data imbalance – i.e., datasets of different sizes – and data heterogeneity – i.e., shifts in distribution– across the datasets. To this extent, we first propose an overview of the significant issues related to the design of a collaborative machine learning system. Then, to aid the design, we propose a framework to categorize various distributed learning approaches in the literature. In the remainder of the study, we propose a set of experiments in which we use different learning approaches to solve common tasks and issues in machine learning applied to medical imaging.

[2] `https://www.rcr.ac.uk/press-and-policy/policy-priorities/workforce/radiology-workforce-census`

1.1 Challenges

1.1.1 Data Availability and Standardization

A significant barrier to the successful application of AI in healthcare is related to data availability. Although healthcare institutions generate massive amounts of data every day, only a fraction of them is actually available for research. This disparity can be related to several reasons, such as the cost of electronic data management [8], the standardization processes, and privacy issues.

Public sharing of healthcare data is a constructive approach for research, as it allows comparative analyses of both clinical and data science approaches. However, building shared datasets requires a standardized approach primarily. In healthcare, Electronic Health Records are the generally adopted standard. However, the user experience related to EHR is affected by a large number of factors that could make it difficult to administrate this kind of data [9], adding complexity to the process of building collaborative systems and datasets. For clinical imaging, the universal standard is DICOM [10], born as a communication standard and nowadays used for storing, exchanging, and transmitting medical images worldwide. The digital storage of medical imaging has enabled the success of AI in healthcare, as a standardized format can provide easier data access from researchers.

1.1.2 Data Imbalance and Heterogeneity

The design of a multi-centric machine learning approach inevitably includes a data acquisition process. When dealing with data coming from multiple different sites, there is a set of issues that can arise. The main issue regarding data is related to the different compositions of the various institutions' datasets. In our work, we discriminate two different sources of data imbalance: *Data Heterogeneity* accounts for the different distribution of data -e.g., diseases, class imbalance-, while *Data Imbalance* is related to the different datasets cardinality - e.g., an institution holding more patient records than the others. In the following paragraph, we provide a more detailed description of the two issues.

We refer to multiple datasets as *heterogeneous* when data in the different institutions are not independent and identically distributed (IID). Data heterogeneity is one of the major challenges for distributed learning, in particular when applied to healthcare [11, 12], as any bias in the data could affect the performances of machine learning models. Heterogeneity can arise from differences in population demographics (skewed data), epidemiolog-

ical distribution (label imbalance), or different data acquisition processes between the institutions (domain shift) that could impact data quality or the feature distribution in the data.

As machine learning models performances are highly dependant on the data availability, a distributed training system should account for scenarios in which more institutions hold different amounts of data, as the unbalance in how each institution is represented in the system can lead to models that fail to generalize when presented samples for the under-represented populations. This represents a type of imbalance that is specific to distributed settings, as in such a scenario, multiple different datasets are collected in a collaborative system. A possible approach to alleviate this issue could be the generation of synthetic data that is similar to the under-represented datasets. For example, in medical imaging, a recently proposed approach uses generative models to produce synthetic patient scans that follow the same distribution that is used to train the model [13]. It is worth noting that such an approach could only be used to balance the representative power of the datasets, as generated data could not possibly introduce new knowledge in the system.

1.1.3 Privacy Issues

A major challenge in healthcare data is related to privacy concerns. Privacy is notoriously difficult to be defined formally. Price and Cohen [14] provide an analysis on privacy issues in medical Big Data. They define two non-exclusive categories in which privacy concerns may arise. Consequentialist concerns "result from negative consequences that affect the person whose privacy has been violated and can have tangible consequences". Deontological concerns "are related to the ethical problem of data ownership and potential loss of control, even in the presence of no direct harm". The collection of medical data is regulated differently in the United States and European Union. The US treats healthcare data differently depending on how data is created and who is handling it. Conversely, EU GDPR sets a general regulation for health data, independently of the format, the collection, or who the custodian is.

Several techniques have been proposed to deal with healthcare research data. The most widely used are *anonymization* and *pseudo-anonymization*. An anonymization technique is the simplest method and consists in removing all information that can be used to identify a patient. However, this process is often not technically attainable, depending on the nature of the data -e.g., for genomic data. Moreover, there's is still debate on what an

acceptable level of anonymization should be [15]. Pseudo-Anonymization is a technique that substitutes sensitive information in the data with synthetic data. The main difference is that the latter technique also allows re-identification by storing a secret key that allows matching the data sample with the patient identity. This, however, poses further challenges with the management of the keys and could represent an issue in case of data breaches or theft.

1.1.4 Security

Distributed Machine learning relies on data exchange between different agents or computational nodes. In an ideal scenario, a Distributed Learning system can be accessed only by the parties collaborating in solving a particular task. However, due to the need for data exchange between different actors, a set of potential security and privacy threats can arise. When considering the healthcare domain, the nature of treated data requires particular care for preserving the confidentiality and integrity of data. While it is reasonable to assume that each party can be cooperative, cybersecurity issues in one or more computational nodes can pose a security threat for the system. Thus, it is important to identify all the actors and the potential threats in a distributed scenario. As deep learning techniques became increasingly adopted worldwide, cybersecurity research developed to consider solutions to possible threats. The possible security threats could target the dataset, for example with re-identification [16] or dataset reconstruction, or attack the machine learning algoritm - e.g. Adversarial Attacks [17], Model-inversion attacks [18]-.

In our work, however, we focused on providing design solutions of a learning system from a Machine Learning perspective. For this reason, we assumed an application context for our framework in which no substantial security threats are present, leaving the possible security improvements as future work. Nonetheless, in section 2.4 we include an overview of the most common techniques that can be applied for securing such a system.

1.2 Objectives and Thesis Organization

The objective of this thesis is to address the typical challenges in Collaborative Machine Learning for Medical Imaging and provide practical and methodological solutions for different use-cases.
In particular, we addressed the following research questions:

(i) What are the main practical challenges that limit the adoption of Collaborative Machine Learning in a Medical Imaging context?

(ii) How can we provide a coherent taxonomy to categorize the different Collaborative Learning approaches present in the literature?

(iii) How different kinds of heterogeneity can be addressed in a collaborative setting?

(iv) How do the two most popular Distributed Learning approaches perform in a data heterogeneity context? What are the design implications of choosing one method over the other?

(v) What challenges can be addressed using transfer learning in a collaborative setting?

(vi) How can we address the specific challenges of Multi-Modality Imaging (e.g., MRI)?

The first two research questions are addressed in Chapters 1 and 2, respectively. The remaining questions are addressed in the following chapters, organized by learning paradigm (Ensemble Learning, Distributed Learning, and Transfer Learning). The summary of each chapter provides a more detailed description of the context and its specific challenges. Thus, the remainder of this work is organized as follows:
In Chapter 1, we propose a structured overview of the most significant challenges in the design of a collaborative machine learning approach to medical imaging.
In Chapter 2, we first propose an introduction to the machine learning methods for medical imaging. Then, we introduce the leading machine learning tasks and their relation with some medical image tasks. Lastly, we introduce our framework of distributed learning approach and use the introduced taxonomy to inspect and compare the relevant contributions in the literature.
In Chapter 4, we use *ensemble learning* to study the settings of *data heterogeneity*, and *model heterogeneity* on a segmentation and a classifica-

tion tasks, comparing different approaches based on ensembles of Convolutional Neural Networks.
In Chapter 5, we propose a comparison of two recently introduced distributed learning techniques in a setting of data heterogeneity, using a Chest X-Ray classification task as a case study.
In Chapter 6, we first study *transfer learning* approaches to exploit the feature extraction pipeline of CNNs to build different kinds of classifiers, then we investigate how to perform *domain adaptation* on a smaller private dataset using our models. Then, focusing on multi-modality imaging, we introduce a segmentation model based on Adversarial Networks and study transfer learning approaches in a multi-modality setting. Lastly, we show how to address the problem of *missing modalities* by performing image-translation and testing our approaches using our segmentation models.
Finally, in Chapter 7, we present our conclusions and future works.

Contributions
This work includes some contributions from our previously published articles. In particular:

- Our distributed learning taxonomy and survey of the state-of-the-art, has been published in 2021 in the European Journal of Nuclear Medicine and Molecular Imaging (Kirienko et al. [19]). In this thesis we present an extended and more technical version in Chapter 2.

- Our experiments on Brain Tumor Segmentation using Adversarial Networks have been published in 2020 International Joint Conference on Neural Networks (IJCNN) (Giacomello et al. [20]). The results are reported in Chapter 6 and applied also in other experiments described in Chapters 4 and 6.

- Our experiments on Brain MRI Generation using GANs have been published in the 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (Alogna et al. [21]) and further expanded in this thesis with additional experiments and considerations in Chapter 6.

- Our experiments on Image Embeddings and Model Ensembling for Chest X-Ray Interpretation have been published in the 2021 International Joint Conference on Neural Networks (IJCNN) (Giacomello et al. [22]), and described in Chapters 4 and 6 of this work. The publication of the additional experiments involving the private dataset described in Chapter 6 is currently ongoing.

- Our comparative analysis on Distributed Learning approaches for Automated Chest X-Ray Diagnosis are currently available as pre-print on ArXiv (Giacomello et al. [23]) and is discussed in Chapter 5.

CHAPTER 2

State of the Art

In this chapter, we first propose an overview of the typical tasks and approaches that are addressed by Machine Learning, with a focus on the context of medical imaging. Then, we introduce Neural Networks and provide a brief overview of the most notable contributions that defined and improved such approach over the years. Then, moving to address the challenges addressed in the previous chapter, we introduce Distributed Learning and provide a taxonomy for the techniques that are included in this field. Lastly, we show and analyze the most relevant distributed learning contributions in the field of medical imaging and healthcare in general.

2.1 Machine Learning Tasks and Approaches

This section provides an overview of the most common applications of Machine Learning tasks relevant to a healthcare scenario. The proposed list does not constitute an exhaustive list, but it is meant to provide a general overview of the most important topics and the practices used to address each task.

2.1.1 Regression

Regression is the problem of estimating the relationship between an outcome variable and one or more covariates or features present in the data. Most commonly, a *linear regression* approach is used to learn a function of the features to produce scalar predictions from new inputs. More formally, a regression model can be described by:

$$Y_i = f(X_i, \beta) + \epsilon_i \tag{2.1}$$

where Y_i is a set of variables described from the data that depends on the dependent variables X_i, β is a set of unknown parameters that have to be learned, and ϵ_i is a set of *error* terms which are assumed to be in the data – but not directly observed –, either due to statistical noise or other variables not taken in consideration. The objective of a regression task is to estimate the function $f(X_i, \beta)$ that better describes the available data and that could be used to predict values of Y for unseen data.

Regression problems in medical imaging are useful to estimate scalar parameters directly from images. Due to the large number of possible applications, presenting an exhaustive overview would be out of the scope of this study. However, we cite two examples that help understand the differences of regression compared with the other tasks explained later in this section when using imaging data. A first example is the estimation of brain age from T1-Weighted Brain MRI. To this extent, Cole et al. trained a deep learning model to predict chronological age in healthy brains, proposing the use of the predicted values as a biomarker related to neurodegeneration and age-associated brain diseases [24]. Regression tasks are often related to evaluating risk, as risk factor usually depends on measures that can be quantitatively estimated. One example is the prediction of osteoporosis and bone fractures, which involves the estimation of bone mineral density (BMD). A possible approach is to build a model that estimates BMD numerically [25, 26, 27, 28], then use the estimated value to calculate the osteoporosis risk [29].

2.1.2 Classification

Classification is a central topic in machine learning, as many tasks can be posed as a classification problem. Given an input sample, a classification problem requires assigning a label, or a class, to the newly received data. In computer vision, classification problems are related to the *pattern recognition* problem, which is the problem of recognizing whether a specific pattern – e.g., a tumoral lesion — occurs in an input image. Indeed, the problem of diagnosing a disease given a clinical record or an input image is a classification problem, as it requires to label the input sample as Positive – e.g., the disease is found – or Negative – no findings –.

An additional categorization of classification tasks is related to the number of classes: A *multi-class classification task* is a task in which each sample has to be categorized into precisely one out of many classes. A further generalization is the *multi-label classification task*, which does not make assumptions on how many classes a sample can belong to.

The most frequently addressed medical imaging problem is the automated diagnosis, which can be accomplished using different kinds of medical images as input, such as MRI, CT, or X-Rays, and aims at labeling each image with a particular label indicating a disease. As we introduce in the following chapters, we address the Chest X-Ray Diagnosis [30] task, which is posed as a multi-label classification problem since more different diseases can be present in the same X-Ray image.

Another example of a classification task is survival prediction from Brain MRIs. Although overall survival can be expressed in years or months, it is a common practice to define classes of survival – e.g., long-survivors, mid-survivors, short-survivors — [31, 32, 33] and to train a model to predict the survival class instead of the exact survival value, which could be misleading and prone to *overfitting*.

In the context of osteoporosis risk prediction, an alternative approach would be to build a classifier that, instead of predicting BMD, predicts whether a fracture occurred or not after a given amount of time – e.g., ten years — [29, 34].

2.1.3 Image Segmentation

Semantic segmentation is a problem related to Computer Vision and has been widely studied in recent years thanks to the adoption of Deep Learning models. Segmentation can be posed as a combination of multiple classification problems, in which every pixel of an image has to be assigned with a class. A typical segmentation problem in healthcare is tumor segmentation: given a medical image such as a Brain MRI, a segmentation model is capable of assigning a different class to the tumor, and the healthy tissues [31]. Other segmentation tasks involving the brain are anatomical brain segmentation [35, 36], White Matter / Gray Matter segmentation [37], and lesion segmentation [38]. A popular application of deep learning in segmentation task is also lung nodule segmentation [39], and in general cancer segmentation involving the abdomen [40]. Sementation is also important for detecting organs in patient images [41, 42, 43, 44], as it allows to develop better and more specialized diagnostic models for each anatomical district.

2.1.4 Image Localization

An alternative approach to segmentation is Image localization – also dubbed *detection* in some works – which is the problem of localizing a specific object or area inside an image. Conventionally, the output of an Image Localization model is a bounding box around the area of the image that represents an object having particular features -e.g., a lesion, a medical device, etc.-. When accuracy at a pixel level is not required, image localization can be used instead of segmentation as manual labeling of images can be made much faster if the annotator has to mark the target object's position instead of defining the precise area. However, depending on the implementation, image localization can often be used together with segmentation, as a common practice is to exploit segmentations to produce a bounding box around the region of interest using computer vision algorithms. In Chapter 6 we show another possible approach to image localization – although not the scope of the study – to localize diseases by exploiting classification models using a GradCAM [45] approach.

While Deep Learning eased the development of segmentation pipelines, Image localization is still widely used in medical imaging [46]. Common tasks include detection of brain sclerosis [47, 48], cerebral micro-bleed [49, 50], breast tumor [51, 52, 53], lymph nodes [48, 54, 55], pulmonary embolism [56], lung nodules [57], bone structures such as intervertebral disc [58] spine fractures [59] and knee cartilage [60], and colon conditions

such as colitis [61] and polyps [54]. Lastly, multi-organ detection is often addressed using localization [62].

2.1.5 Image Translation

Image translation is a Computer Vision field that has been recently introduced in healthcare research. The problem of image translation consists in producing an image in a *target domain*, given the corresponding image in the *source domain*. For example, it is possible to train a machine learning model to produce a given MRI modality given the same MRI in another modality, as we address in chapter 6. This approach has received increasing interest thanks to the popularity of GANs and the contributions of Phillip Isola et al. [63] that developed a general-purpose architecture, known as Pix2Pix.

As we address in Chapter 6, the image generation task is tightly related to the problem of missing data and data heterogeneity. In medical imaging, differently from other imaging fields, an image is often composed of several different modalities, which captures different physics phenomena – e.g., in MRI imaging. A relevant issue with MRI images is that often not all the modalities are available for each patient. This generally happens for several reasons, such as prohibitive scan times and costs, artifacts, data corruption, acquiring machine settings, adverse reactions to contrast agents, etc. When not every modality is available for all the patients in a dataset, this can be an issue if the available model –e.g., classifier— requires a fixed set of modalities as input. To this extent, Sharma and Hamarneh [64] designed a multi-modal architecture where the missing modalities are considered as zero-valued inputs. Another possibility is to exploit Image translation to generate the missing modality from the available ones. Dar et al. used a *Cycle GAN* [65] to generate missing modalities in a uni-modal setting [66], Yu et al. [67] generated 3D FLAIR images generated from T1 and later used to train a classifier together with T1 images, which led to a performance improvement. Similarly, in [68] the authors addressed the problem of missing FLAIR sequences in *white matter hyper-intensity segmentation* task by generating the FLAIR images from T1 images while performing the segmentation at the same time. Finally, Ge et al. [69], addressed the problem of missing modalities using pairs of GANs; given two modalities for which there are missing samples (e.g., T1 and T2), a pair of generators are trained to produce one modality from the other (e.g., G_a: T1 $\rightarrow$ T2, G_b: T2 $\rightarrow$ T1). The corresponding discriminators use the input of the other generator as the ground truth (e.g., $D_a(T2, G_b(T1))$ and $D_b(T1, G_a(T2))$

respectively).

Another common approach to deal with missing modalities consists of training a model invariant to the input contrast modalities. To this extent, Havaci et al. [70] computed a latent vector for each input modality and then combined all the latent vectors into a single representation that accounts for every modality in input. This method was later extended in [71] to allow unlabeled inputs, i.e., to no longer specify which modalities are provided as input. Finally, in [72] the authors exploited Variational AutoEncoders (VAE) [73], to train an encoder for each modality and, at the same time, to generate the missing modalities.

2.1.6 Neural Network Interpretation

One of the significant drawbacks of CNNs is that they are considered black-box models, as they are hard to interpret by humans. This fact represents an issue for their adoption in critical fields, such as medical imaging. The trustworthiness of AI models for clinical diagnosis and prognosis has to be accurately assessed before their applications in a real setting. To this extent, the U.S. Food and Drugs Administration proposed in 2019 a framework to regulate the Artificial Intelligence and Machine Learning Medical Devices, defining a possible approach to enable administrations and manufacturers to evaluate and monitor AI-based software products during development and monitor postmarket performances [74].

In the latest years, several algorithms have been proposed to overcome the problem of neural network interpretability. DeepLIFT [75] and SHAP GradientExplainer [76] are based on feature importance and aim to measure the importance of each input feature in the predictions by using the coefficients of linear models used as interpretability models. Another approach is DGN-AM [77]: it evaluates which neurons are maximally activated concerning a particular input observation aiming to find input patterns that maximize the output activation.

Other similar and widely used algorithms are CAM [78], Grad-CAM [79], and LRP [80], which create coarse localization maps of the critical regions of the input defining the discriminative regions for a specific prediction.

In our work we focused on the CAM and Grad-CAM algorithms, which will be briefly introduced. In particular, the CAM algorithm requires a *Global Average Pooling* layer to be inserted between the last convolutional layer and classification layer of the network. For every feature map $f \in 1, ..., n$ of size $(K_f * K_f)$ produced by the convolutional layer, the GAP layer computes the average of the feature map over the spatial dimen-

sions, producing n different scalars. Lastly, for each output classes a set $W_1, ..., W_n$ of weights is learned to map each Global Average to the final class score. To compute the Class Activation Maps, the CAM algorithm perform a linear combination between the weights $W_1, ..., W_n$ of a chosen class c and the corresponding feature maps, resulting in an heatmap that represent the areas of the input that contribute to the network prediction for the class c.

The Grad-CAM algorithm instead uses the Gradient of the network output to perform localization. In particular, for a chosen class c, the *logit* of the class is derived with respect to the feature maps of a convolutional layer, then they are averaged using a GAP layer, obtaining a set of n *importance weights*. Lastly, similarly to CAM, the *importance weights* are linearly combined with the original activation maps and a *ReLU* activation function is applied to only consider positive influence.

The generated CAMs for both methods are thus heatmaps of the same size of the considered feature maps. For this reason they usually need to be rescaled up to the input size before visualization.

Interpretation and Explaination of Neural networks are still an active field of research. Degrave et al. [81] recently demonstrated that Deep Learning models for COVID-19 detection relied on spurious shortcuts such as laterally markers, image annotations, and borders to distinguish between positive and negative patients, instead of identifying real markers of COVID-19 in the lung field. They claim that explainable AI (XAI) models should be applied to every AI application in medicine and should be a pre-requisite to its clinical translation to routine.

2.2 Machine Learning and Neural Networks

2.2.1 Neural Networks

Machine Learning is a vast field of AI that includes several methods such as Support Vector Machines (SVM), Decision Trees, Random Forests, Clustering algorithms, and Neural Networks. Up to the first decade of the 2000s, the typical process to classify an image required a feature extraction step prior to training a machine learning model, as the majority of methods in use were either unsuitable or not efficient in treating such kind of data. This required the development of ad-hoc algorithms and methods to extract features that were relevant for the problem manually, then train a machine learning model –e.g., based on SVM or Neural Networks — that solved the task.

In our study, we focus mainly on Neural Networks. In 1958, Rosenblatt introduced the *perceptron*, a kind of linear binary classifiers that became the elementary unit for *feed-forward neural networks* [82]. The perceptron was intended as an image recognition machine rather than an algorithm. In the first implementation, a set of photocells were connected to the perceptrons, and the weights were set using potentiometers and electric motors [83]. However, a single layer of perceptrons could not be trained to discriminate even simple kinds of patterns such as the XOR function. This led to stagnation in research, which delayed the introduction of neural networks for many years. In the 1980s, the backpropagation algorithm [84] were used to train multi-layer neural networks [85], which led to a new increase in interest in Neural Networks; however, the computing power needed for training such models was still prohibitive for a successful application on a large scale.

A neural network is thus a mathematical model composed of a set of *artificial neurons*, organized in one or more layers. Each neuron is connected to all the neurons of the previous layer, as shown in Figure 2.1. Each connection is mathematically represented by a *weight* that is the variable to be optimized during the learning process. Given a layer of neurons, the output of a neuron k is given by:

$$y_k = \phi(\sum_{j=0}^{m} w_{kj} x_j) \tag{2.2}$$

Where ϕ is a suitable *activation function* such as *sigmoid*, *tanh* or *ReLU*[86], $x_0, ..., x_m$ are the outputs of neurons of the previous layer and w_{km} are the weights associated with the connections between the neuron m of the pre-

vious layer and the considered neuron k. Often x_0 is set to $+1$ and the corresponding weight w_{k0} is set to a *bias* b_k.

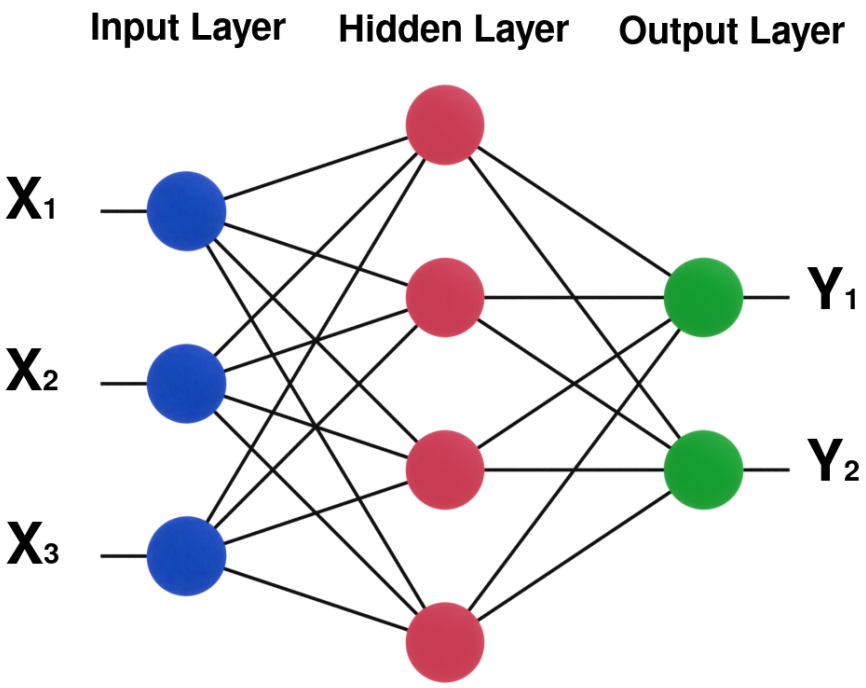


Figure 2.1: *A Feed-Forward Neural Network with 3 inputs x_1, x_2, x_3 and 2 outputs y_1, y_2. Each neuron, represented by a node, is connected to all the neurons of the previous layer. All the layers that are not the input nor the output one are called* hidden *layers – shown in red — and allow the network to model more complex functions.*

2.2.2 Convolutional Neural Networks

In 1998, LeCun et al. proposed LeNet [87], which achieved the first successful application of Convolutional Neural Networks (CNN) in handwritten document recognition, using a 7-layers network. Convolutional Networks differ from standard Neural Networks in the fact that they use a set of *convolutional layers* to perform feature extraction directly during the training process using the *convolution* operator. In this kind of layer, the weights are organized as a set of *filters* (or *kernels*) of size KxK that represent features in the image. Generally, K is chosen to be small, typically 3, 5, or 7, while a large number of different filters is used. A high-level overview of a convolutional layer is shown in figure 2.2. By cascading more con-

volutional layers, it is possible to capture features at different scales and levels of abstractions. Typically, the first layers capture simpler concepts as edges and corners, while layers closer to the output capture more complex concepts.

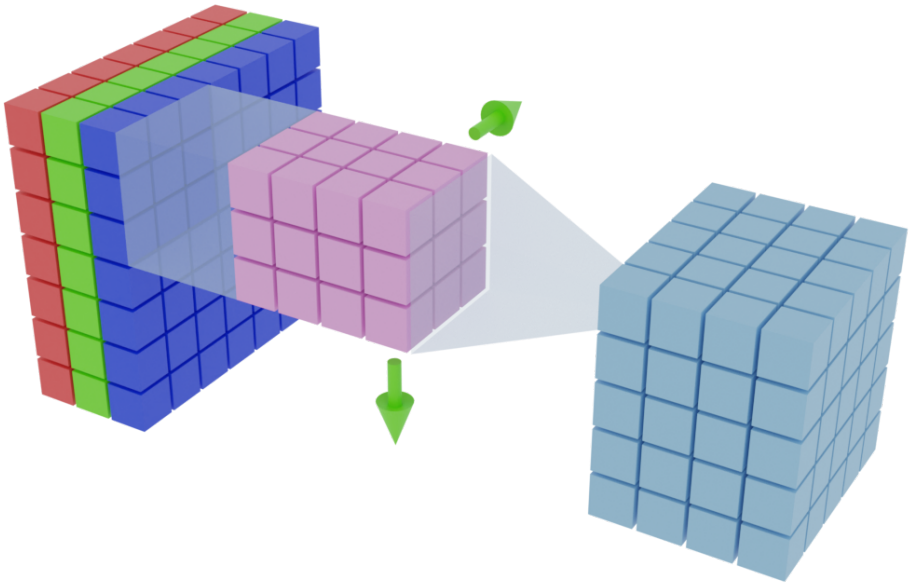


Figure 2.2: *Overview of a Convolutional Layer. From left to right: (i) a RGB representation of an image, (ii) the convolution operation is performed using a set of $N = 4$ filters having dimensions $KxKxN$ – $3x3x4$ in the figure – by sliding the kernel across the spatial dimensions of the image – indicated by green arrows —. (iii) The result of the convolution at each spatial location is a vector with the same number of values as the number of filters. Thus, a convolutional layer is able to capture* local features *in the image space – i.e., the result of the convolution at each pixel considers a set of KxK pixels in the input, which are assumed to share some kind of feature.*

While CNNs existed for decades, it is only in the first years of the 2010s that researchers regained interest in neural networks, mainly thanks to the technological advances that allowed to use of GPUs to speed-up neural network computations and the increased availability of image data. In 2012, Krizhevsky et al. designed Alexnet [88], a CNN which was inspired by LeNet but considerably bigger and with multiple layers. AlexNet won the ImageNet 2012 competition, in which the proposed algorithms are asked to classify every image in a dataset among 1000 classes. This result highlighted the effectiveness of Convolutional Neural Networks in solving Image Analysis and Computer Vision tasks, determining the success of the

Deep Learning field. Deep Learning focuses on methods that stem from the field of Neural Networks. The adjective "Deep" indeed refers to the massive amount of artificial neuron layers that are used in these approaches, in contrast with the "shallow" neural networks commonly used in Machine Learning. CNNs have soon become a widely adopted model since their versatility in many fields that involved imaging. An example of the ability of CNNs to learn features that are useful for different tasks is given in [89], where the authors follow a *multi-task learning* approach to demonstrate that it is possible to develop a single model to detect brain, breast, and cardiac jointly.

2.2.3 Feature Extraction

The major advantage of Deep Neural Networks concerning other Machine Learning approaches is that the deep structure allows capturing more complex and higher-level patterns directly from the data. As a notable example, in 2012, Le [90] created a Deep Neural Network capable of learning various concepts from images, such as human faces or cat faces, without the need of knowing whether each image contained the concept or not. This capability of learning high-level concepts from the data itself is one of the strengths of Deep Learning: while other Machine Learning algorithms require a separated feature extraction phase to be successfully applied, Deep Learning algorithms can learn an effective data representation automatically. Furthermore, data representations learned from data are generally helpful also for related tasks with limited effort [55], whereas feature extraction is usually a problem-dependent and time-consuming task.

While for general-use images, a completely automated feature-extraction approach is both beneficial and desirable, machine learning models used in medical imaging must be interpretable by humans to be trusted by physicians and patients. The complex interactions between weights and functions in a CNN make the interpretability of the results difficult for the user. For this reason, in medical imaging, a field of research called *radiomics* focuses on extracting features from imaging data. Radiomics is a field that emerged from radiology and oncology, which is designed to develop decision support tools. Instead of considering the images as data that are intended to be viewed by a human, radiomics establishes a process of analysis that extract a large number of quantitative features from medical images, and subsequently mining the data to find patterns and patients characteristics that cannot be easily noticed by visual inspection. This feature extraction process is precious both for research and clinical practice, as

it produces features that allow the development of better decision support systems [91]. Radiomic features can be classified in *semantic features*, – i.e., features that are commonly used by radiologists to describe lesions – and *agnostic features* – i.e., features that pertain to quantitative descriptors derived from statistical descriptors–.

The interaction between radiomics and Deep Learning is an open field of research [92, 93]. While our studies focus mainly on the latter, we envision that contamination between the two fields could be beneficial to develop decisional supports systems that are both effective in extracting relevant features from raw data and interpretable.

2.2.4 Generative Adversarial Networks

One last neural network model that is worth introducing is the *Generative Adversarial Network* [94]. This technique – which should not be confused with *Adversarial Learning*, which refers to another approach – involves training a model that is composed of two different neural networks that are trained in an alternating fashion. The first, dubbed *generator*, is trained to generate samples whose distribution is similar to the one found in the dataset. A second network, dubbed *discriminator* or *critic*, is trained instead to discriminate between actual data coming from the dataset and fake data produced by the generator. During the training process, the two networks are playing a zero-sum game in which the generator is trained to trick the discriminator into misclassifying the fake samples, while the discriminator is trained to improve its performances in classifying real and fake examples. As the training process proceeds, the samples produced by the generator become increasingly similar to real data, being indistinguishable from it when the optimum should be reached. Radford et al. [95] improved GANs to work more effectively with larger dimensionality of data; instead, Mirza and Osindero introduced Conditional Generative Adversarial Networks (cGANs) [96], that extended GANs by conditioning the input of the model with additional data, which could be used to direct the data generation process. However, this kind of model is currently an active field of research, and several other variants are available. In particular, cross-modality image synthesis – i.e., the conversion between different modalities of the same image – is one of the main applications of this architecture to medical imaging. A survey published in 2018 [97] collects the major contributions in this field through the application of GANs. Outside of medical applications, GANs gained overwhelming popularity thanks to the successful applications in face generation [98], photo restoration [99],

image semantic synthesis [100] and many other fields. To generate data, GANs use a noise vector called *latent space*, which can be either explicit as a network input or implicitly generated by randomly disabling some network weights [101]. However, in a medical setting, generating synthetic data that is similar to the available dataset in an uncontrolled fashion has limited applications. For this reason, other use of GANs in medical imaging are related to *image segmentation* and *image translation*. To this extend, the generator has to be modified to receive as input a sample of real data – either the image to segment or an imaging modality that has to be translated – and trained to produce the desired result. By exploiting the same adversarial mechanism, the GAN can be used to solve new kinds of tasks that go beyond the synthetic generation of images.

Lastly, it is worth mentioning the drawbacks of a GAN approach. Due to the adversarial process, GANs are notoriously difficult to train [102] as their convergence properties are not stable as those of a standard neural network, and they may fail to converge. Moreover, for a *vanilla* GAN, the value of the loss function is no more an indication of the training progress and it is not monotonic. In addition, since two – or more – networks have to be trained simultaneously, the computational resources needed for this kind of approach are generally higher than those of a standard CNN.

2.3 Distributed Learning

The problem of training models capable of aggregating knowledge from different data sources - e.g., different hospitals or research centers- without physically exchanging data among the participants can be solved according to different perspectives and levels of abstraction. This poses the issue of defining a new taxonomy for classifying the approaches that belong to distributed learning.

In defining this taxonomy, we first defined three main categories that can be used to describe all the proposed distributed approaches in the literature: (i) Ensembling methods, (ii) Split Learning, (iii) Federated Learning. These three categories are pertinent to distributed learning, as opposed to *Centralized Learning* and *Local Training* that represent the two extremes in which data is completely shared -i.e., all local datasets are merged in a single one- or no communication between the nodes occurs, respectively. Previous works in the literature proposed to classify the different distributed learning methods based on computational principles such as data parallelism, and communication topology [103]. Here, we focused on two more general design principles: (i) how the model parameters are displaced over the network of nodes, and (ii) how the nodes interact in the training and inference process. These two dimensions are the most relevant choices to make to design a distributed learning system that best suits the requirements in terms of data privacy, technical constraints, and operation workflow.

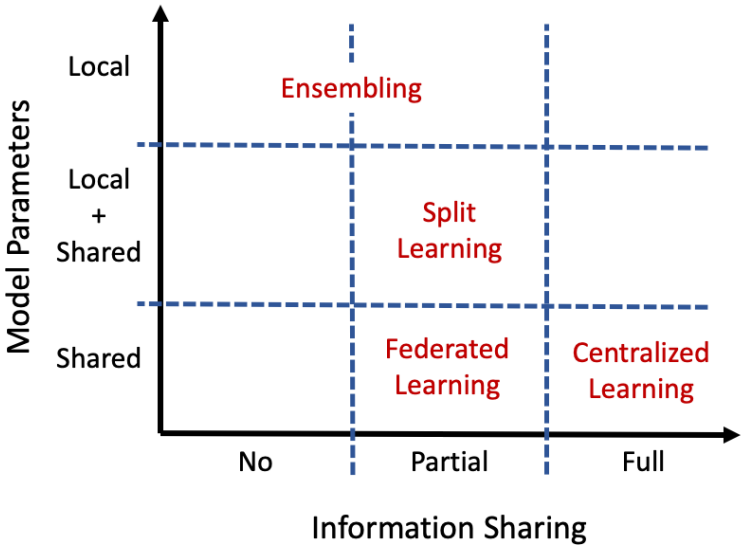


Figure 2.3: *Overview of Distributed Learning Paradigms*

Figure 2.3 shows how the most popular distributed paradigms differ according to our design principles. The displacement of model parameters affects how many models we can obtain at the end of the training process: (i) All the parameters can be shared among the nodes, resulting in a single model for the node network, (ii) all the parameters can be local to each node, resulting in a local model for each node, (iii) some parameters can be local to each node and some others can be shared, resulting in *hybrid models* such Split Learning. For each method, also information sharing between the nodes can be different for each paradigm: Centralized Learning approaches have full information sharing since the data is merged in a single dataset, while in distributed learning approaches, data sharing can be avoided entirely or can be only partial - e.g., exchanging only model parameters or their updates-.

Each distributed learning paradigm accounts for the displacement of data and models across the nodes and their interaction during the training and inference phases. Two of the main factors crucial to determining the distributed architecture are the task to address and the operating context. Throughout this chapter, we introduced an overview of the tasks – both from a machine learning and applicative standpoint – that can be addressed using a distributed learning paradigm. Each distributed learning paradigm differs from the others by three fundamental aspects:

(i) Number of Output models: Depending on how the learning procedure is designed, each approach's outcome can be either a single or multiple models. When the final artifact is a single model, each client can receive an identical copy of the model to use locally. On the other hand, using an approach that produces different models has the advantage of task specialization - e.g., multi-task learning - or client personalization according to each participant's needs.

(ii) Weights Ownership: Each distributed paradigm and learning procedure involves different displacements of the model parameters. This dimension's two extremes are the scenarios in which the model parameters are entirely distributed or completely centralized. If a learning system uses only distributed parameters, each node acts upon a locally stored model, and no centralized model exists. In centralized ownership settings, each node receives a copy of the parameters and sends the updates back to a server. Finally, the server stores the current global state of the model. A combination of these settings is also possible, as in Split Learning: a subset of the model parameters is stored on a master server. The clients own another shard of parameters.

(iii) Nature of Exchanged Data: In every distributed learning setting,

some information exchange degree is unavoidable. This aspect is essential to privacy concerns and the networking performances of the proposed method. For example, some paradigms only share a pre-trained model, while others require sharing the model outputs or the parameter updates.

2.3.1 Ensemble Learning

Ensemble Learning is a widely studied family of methods [104, 105] that allow combining multiple machine learning models to produce a single model, which generally shows better performances than the single ones [106, 107]. Ensemble Learning has been investigated in the field of supervised learning since the late seventies, and since then, many different algorithms and techniques have been studied [108]. In the context of distributed learning, this kind of approach can be used to train a local model for each node using its private dataset. Once the training is done, these models can be combined in several ways, for example, by simply averaging their output, defining custom aggregation methods as we present in chapter 4, up to training a meta-model on the local models outputs to produce a final prediction (Stacking [109]). Ensembling methods do not require sharing training data during the training phase since the aggregation phase happens after the training of local models is ended. The only exception is data needed to train the meta-model if stacking is used or training statistics -e.g., training accuracy, models error, etc.- depending on the chosen aggregation method. On the other hand, as the training does not produce a single model for all the node networks, using ensembling implies either sharing the local models among all the network nodes or sharing the new data for which prediction is needed. This paradigm makes it possible to use any machine learning model, as ensembling relies exclusively on locally trained models. It also allows the use of different models on each node, which might be very convenient when data is heavily heterogeneous. Figure 2.4 shows an overview of how ensemble methods work.

2.3.2 Split Learning

Split Learning is a class of methods that involve holding different shares of the same model across the network nodes [110]. Due to their layered nature, deep neural networks are particularly suitable for this kind of approach. The idea behind Split Learning is to distribute the layers of a single neural network amongst different institutions. In the simplest configuration, given a neural network composed of N layers, then each *client* institution can host the first $i < N$ layers, while a server hosts the remaining $N - i$.

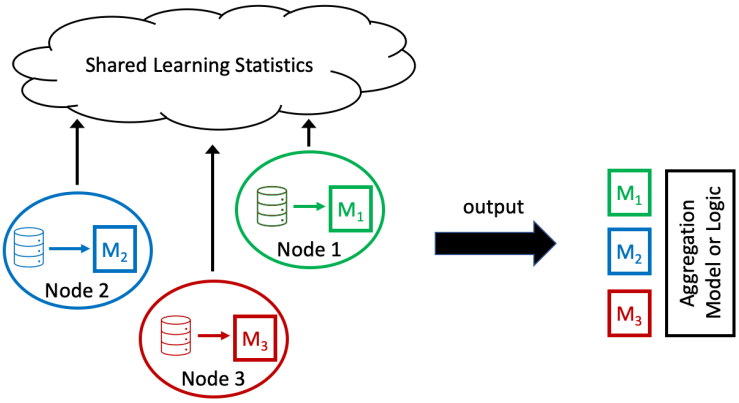


Figure 2.4: *Overview of Ensembling Methods: A local mode is trained on each local node and only some learning statistics are eventually shared. The local models and an aggregation logic (or meta-model) are provided as the final output.*

At each training step, the client computes the forward pass of its section of the network and sends the output to the server, which in turn completes the forward pass and then returns the gradients back to the client. This way, the server never sees any training data, as it allows building a common model by only exchanging extracted features and error gradients between client nodes and server nodes. In 2018, Vepakomma et al. [111] proposed several Split Learning configurations by organizing the network layers among the nodes, providing different solutions concerning data privacy and task constraints. In chapter 5 we investigate the performances of some of these configurations on the task of Automated Chest X-Ray Diagnosis. Figure 2.5 shows an example of split learning used to train an artificial neural network.

2.3.3 Federated Learning

Federated Learning is a class of approaches that involves collaboratively training a single model [112]. The most general federated learning approach involves a shared model that is kept updated during the training process. Thus, each node trains the shared model using its own private dataset and only shares back the model updates. Different training algorithms can be used to distribute the computation among the nodes, depending on both the machine learning model used and the constraints on node communication (e.g., how often the nodes can communicate). Thanks to the generality of the paradigm, many works refer to *Federated Learning* to indicate slightly different approaches to the problem of distributed learn-

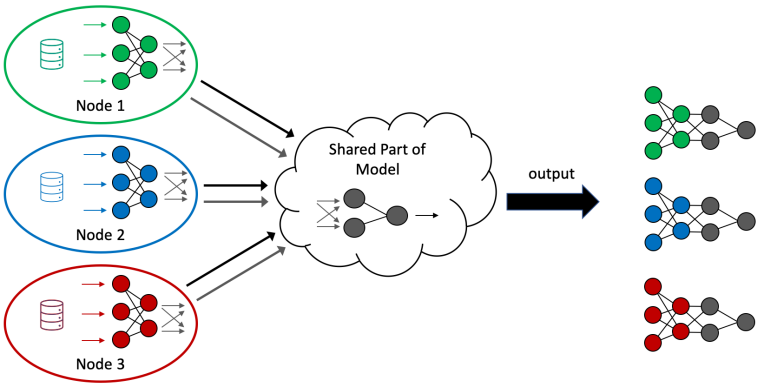


Figure 2.5: *An example of split learning applied artificial neural networks(NNs). The first layers of NN are local and trained only with private datasets. The final layers are instead shared among the nodes and trained with all the data. Each node shares the intermediate output of the NN along with labels to allow the training. A local/global model is provided as output for each node.*

ing: until recent years, distributed learning techniques mainly focused on data and model parallelism to increase computational efficiency. In 2012, Google proposed DistBelief, a framework to train deep neural networks on a large scale, by employing hundreds or thousands of machines. In 2015, Shokri and Shmatikov [114] proposed a distributed version for the Stochastic Gradient Descent [114] that allowed multiple clients to train neural networks parameters hosted in a *parameter server*. Thanks to the diffusion of increasingly powerful mobile devices, the focus of distributed learning approaches moved toward the possibility of training large models directly on smartphones [115]. This innovation posed the challenge of preserving the users' privacy, which is extremely relevant in a healthcare setting. In 2017, Google further defined the concept of Federated Learning [116, 117] by formally describing the Federated Learning problem. As stated in [117], the main difference between Federated Learning and the previous distributed learning approaches arises from the need to address a high number of clients that are highly unbalanced, have poor availability, and hold non-i.i.d data. In a healthcare scenario, the number of clients is usually limited, and their availability is higher than that of a smartphone; however, the other constraints have high relevance for clinical data.

For the reasons above and for providing a coherent taxonomy, we dub as Federated Learning all the distributed approaches in which the system is composed by a server and several clients, characterized by the following training round:

(i) A subset of clients is selected to download a current state of the model
(ii) Each client updates the model based on their local data
(iii) The model updates are sent from the clients to the server, either in the form of gradients or updated weights
(iv) The server aggregates the models (e.g., by averaging) to improve the global model.

In the next section, we will introduce other distributed learning techniques that can be inscribed under the family of Federated Learning algorithms.

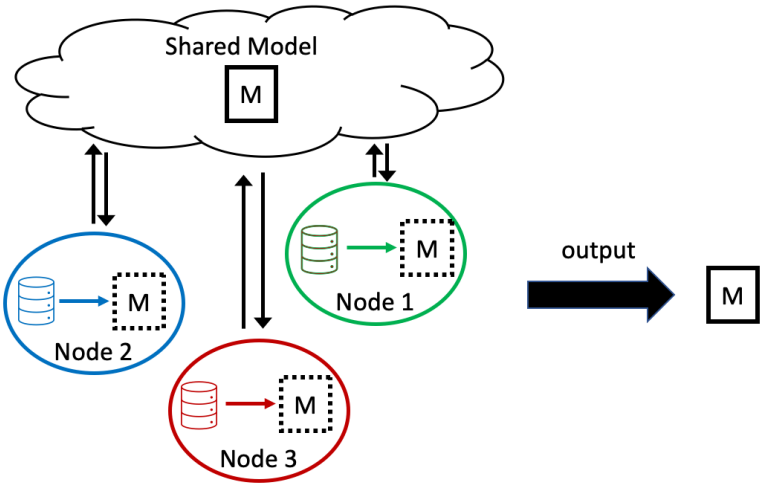


Figure 2.6: *Overview of Federated Learning methods*

2.3.4 Other Paradigms

A popular technique for training privacy-preserving models for healthcare is Incremental Learning (IIL). This technique can be seen as a collaborative extension of Transfer Learning. In this section, we explore the relationship between the two paradigms and how they relate to our framework.

Transfer learning is a general machine learning paradigm that consists in adopting a pre-trained model during the training phase instead of training a model entirely from scratch. If the model has been pre-trained on data that is sufficiently similar to the target dataset, then the knowledge learned from the source dataset can be exploited in the new setting. This technique is generally adopted in training Deep Neural Networks on images, as they require a large number of samples. Another advantage of this method is that it can give good performances when a low amount of data is available, compared to training the model from scratch.

While Transfer Learning is not strictly considered a Distributed Learning method, we included it in our work due to its massive adoption in the literature. For the scope of this work, we consider Transfer Learning a scenario in which each node receives a pre-trained model and autonomously trains it on its own private dataset. This differs from the other learning paradigms introduced in this chapter by the fact that each client does not collaborate with other nodes in the network. The biggest advantage of transfer learning is the personalization of the model, as each node can tailor the final model to their autonomy. The control that the node has on the models allows it to perform re-training or fine-tuning whenever necessary, as it doesn't require defining a protocol between the participants. However, this benefit comes at the cost of not receiving potentially valuable information from the private data stored in other nodes.

Incremental Learning - also referred to as *Cyclical Weight Transfer* [118, 119] - is the most straightforward distributed learning strategy, as it can be viewed as a collaborative extension of Transfer Learning. In IIL, a single ML model is trained iteratively in each node on its local data before sending the model to the next institution. Therefore, the main difference from Transfer Learning is that only a single model exists in the system at a given time, and each client can choose its own different training method to fine-tune it. In our work, we don't make a distinction between IIL and Cyclic Incremental Learning (CIIL) [120]. CIIL is a more general setting in which each client trains the model for a given number of epochs before sending it to the next node, and the training rounds can be more than one. Thus, we consider the number of epochs and the number of rounds to be hyper-

Paradigm	Output Model	Weights Ownership	Exchanged Data
Ensemble Learning	Multiple	Local	Model Output
Split Learning	Multiple	Local + Centralized	Layer Outputs, Updates
Federated Learning	Single	Centralized	Updates/Weights
Transfer Learning	Single	Local	Pre-Trained Model
Incremental Learning	Single	Local	Weights
Peer-to-Peer Learning	Multiple	Local	Updates

Table 2.1: *A detailed description of Distributed Learning Techniques. The second column shows whether the technique produces a single model for the whole network or multiple models. The third column shows whether the weights are hosted by the client nodes or a central server. The fourth column shows the minimum data that is needed to exchange for performing training.*

parameters of the learning paradigm, and we refer to the method simply as Incremental Learning. Moreover, since IIL consists in training a single model over the network of nodes, it can be seen as a special case of Federated Learning, in which model parameters are exchanged instead of their updates.

In [118], the authors claim that a single iteration of incremental Learning can give different results depending on the order in which the institutions are selected. For this reason, incrementing the number of iterations could be beneficial. Moreover, performances of (Cyclic) Incremental Learning compared with Ensemble Learning varied depending on the dataset. One major disadvantage of this method is the problem of catastrophic forgetting [121, 122], in which the model "forgets" the progress made from earlier institutions when trained iteratively on different data. A possible solution for this issue could be to tune the number of training epochs at each node [120].

Finally, Peer-to-Peer learning [123] is instead an alternative method similar to Federated Learning that doesn't require a central server for hosting the most recent model, is based solely on the cooperation between the different clients. Table 2.1 shows a detailed view of the techniques cited in this chapter.

2.4 Security Techniques

In chapter 1.1 we introduced the challenge of securing the computation during the distributed training process. Although in this work we focus on machine learning techniques, we present in this section an overview of the main security techniques that can be applied when implementing a distributed learning system for healthcare.

2.4.1 Differential Privacy

Differential Privacy [124] is a technique formulated in 2006 to secure statistical databases. This strategy consists of preserving the statistical distribution of a dataset while minimizing the amount of recognizable data. In 2016, Abadi et al. proposed to utilize this mechanism in Deep Learning models [125]. In particular, the authors outlined a differentially-private SGD algorithm, which allows applying differential privacy during the training phase of a neural network. The algorithm bounds each example's influence on the gradient loss by performing a clipping operation at each step. Then, privacy is attained by adding stochastic noise to the gradient. Thus, Abadi et al. proposed a method for applying DP to the training algorithm. However, different DP implementations exist, and they can be categorized in input perturbation, output perturbation, exponential mechanism, and objective perturbation, depending on the different approaches of adding noise [126]. A disadvantage of this approach is that the perturbation that DP adds imposes a trade-off between privacy level and predictive capabilities. In their recent work, Choudhury et al. study the case of Differential Privacy in a Federated Learning scenario [127], analyzing the trade-off between privacy and predictive capabilities on two different medical datasets.

2.4.2 Homomorphic Encryption

Homomorphic Encryption (HE) is a kind of encryption that allows the users to perform non-polynomial operations directly on the encrypted data. Thus, a machine learning system that implements Homomorphic Encryption can perform both training and inference without first decrypting the input data. The operations applied to the encrypted data are formalized as arithmetic or boolean circuits. Moreover, a $C-$*evaluation scheme* is defined on a set of circuits C as a tuple, including the generation, encryption, evaluation, and decryption algorithms. Depending on the properties of the circuit and evaluation scheme, HE algorithms can be classified in four levels [128], ranging from *somewhat homomorphic encryption scheme* (SHE) to

fully-homomorphic encryption scheme (FHE). The problem of developing a homomorphic encryption traces back to 1978 [129] and FHE is considered one of the most promising classes for machine learning and cloud systems for arbitrary computation on encrypted data [130]. However, the main limitation of this approach is related to its significant computational overhead. Many FHE schemes have large overhead - although still polynomial - which makes homomorphic computation impractical. Nonetheless, even if the theoretical view of FHE points toward the maximization of the functions that can be applied to encrypted data, it is possible to reach a reasonable trade-off in terms of computational time by choosing an implementation that restricts the choice of the applicable operations. In recent years, homomorphic encryption has been applied with many machine learning techniques like Neural Networks [131, 132], showing promising results in terms of computational efficiency and accuracy. This approach is particularly promising in the field of healthcare: working on this, Wood et al. propose an overview on how to apply fully homomorphic encryption to different tasks in machine learning and bio-informatics [133].

2.4.3 Secure Multiparty Computation

Secure Multi-Party Computation (MPC) [134] is a computational model designed in the early 1980s that focuses on ensuring privacy among the participants of a computation involving different actors rather than external threat agents. An MPC aims to define a protocol for allowing different participants to jointly compute the value of a public function over their private data while keeping their own private data secret from the other participants. Only in the 2000s algorithmic improvements and computational costs reach a level that allowed the implementation of this model in a general-purpose computation system [135]. Other notable applications of MPC include secure auctions, and Secure Electronic Voting [136]. In 2017, Liu et al. proposed an approach for enabling neural networks to perform oblivious model inference[137]. This technique allows a client to request a prediction from a remote machine learning model hosted on a server, preventing the server from accessing requests' data and the client from accessing the machine learning model. The proposed approach uses a combination of MPC and Homomorphic encryption. While MPC could enable a set of computational nodes to perform secure machine learning training, the typical dataset sizes used for this task pose a challenge to Secure Multi-Party Computation. Nonetheless, MPC can be combined in hybrid approaches with other techniques, like Homomorphic Encryption[138] or customized

protocols [139].

2.5 Distributed Learning in Healthcare

This chapter presentst a survey on state of art related to distributed machine learning systems in healthcare. We also included contributions that are not focused only on medical imaging but also on other kinds of data such as Electronic Health Records (EHR) since the techniques applied in one data domain can provide useful insights for other domains such as images.

To present the various contributions, we considered several aspects discussed in this chapter. The first dimensions of analysis are shown in Table 2.2 and are related to the data type, the datasets used in the study, the domain of application, the typology machine learning task, and the kind of machine learning technique used – either the kind of model or the specific architecture in case of neural networks —.

Then, in Table 2.3 we classified the contributions according to the Distributed Learning paradigm used and whether the authors applied privacy ensuring algorithms. We also indicate in this table the kind of metrics that have been used to assess the results. Depending on the focus of the study, either statistical metrics or other indicators such as the number of iterations or the required bandwidth have been used to perform the assessment.

In a distributed learning setting, an important aspect of evaluating an approach is how it performs according to different data settings. In particular, Table 2.4 shows whether data imbalance – i.e., differences in the number of samples between the local datasets – or data heterogeneity – e.g., differences in data distribution across the datasets – have been considered. Moreover, we report if the contribution performed analysis using simulated data. The use of simulated data may be beneficial for assessing how a distributed learning paradigm performs in the presence of data imbalance or heterogeneity, as it allows to mitigate the problem of data availability.

Lastly, in Table 2.5, we report for each contribution the different distributed learning paradigms that have been compared if the analysis has been performed. For categorizing the contributions, we applied the distributed learning framework we presented in this chapter. Moreover, we also show what independent variables have been used to assess the performances of the method and the comparison between multiple paradigms. Lastly, we show the number of nodes that have been used in the experiment.

2.5.1 Comparison between Distributed Learning Paradigms

Remedios et al. performs an analysis of a collaborative model based on In-

Ref.	Data Type	Dataset	Domain	ML Task	ML Techniques
[127]	EHR	LCED, MIMIC-III	Adverse Drug Reaction, Mortality Prediction	Classification	Perceptron, SVM Logistic Regression
[140]	EHR	MIMIC-III	Patient Similarity	Representation Learning	Hash Functions
[141]	EHR	Boston Medical Center	Hospitalization	Classification	SVM
[142]	EHR	UPHS	Drug and Fetal Loss relationship	Regression	Logistic Regression
[143]	EHR	eICU	Mortality, ICU stay time	Classification, Clustering	CBFL, Autoencoders
[144]	EHR	Multiple	Dyspnea Prediction	Classification	Bayesian Networks
[145]	EHR	Multiple	Dyspnea Prediction	Classification	SVM, ADMM
[146]	EHR	Maastro, Michigan univ., The Christie	2-years survival	Classification	Bayesian Networks
[147]	EHR	Multiple Private Datasets	2-years survival	Classification	Logistic Regression
[148]	EHR	Cerner Clinical	Hospital Stay prediction	Regression	Linear Regression
[148]	Numerical	UCI Student Performances,	Student Grades	Regression	Linear Regression
[148]	Numerical	Car Fuel Consumption Data	Fuel Consumption	Regression	Linear Regression
[120]	FLAIR MRI Images	BraTS 2018	Brain Tumor	Segmentation	U-Net
[149]	T1 MRI	Multiple	Brain structural relationship across diseases	Dim. Reduction	ADMM
[119]	CT	VUMC, CNRM, UMD, VCU	Hemorrage Segmentation	Segmentation	U-Net
[150]	CT	NIH, VUMC	Hemorrage Segmentation	Segmentation	U-Net
[118]	Mammographies	DDSM	Lesion diagnosis	Classification	ResNet34
[118]	Retinal images	Kaggle Competition	Retinopathy diagnosis and grading	Classification	ResNet34
[118]	Photos	ImageNet	Image Classification	Classification	ResNet34
[151]	Retinal images	DRC	Retinopathy diagnosis (binary)	Binary Classification	ResNet34
[151]	FLAIR MRI	BraTS	Brain Tumor Diagnosis	Binary Segmentation	U-Net
[151]	FLAIR MRI	CheXpert	Lung Disease Diagnosis	Multi-Label Classification	DenseNet121
[151]	Muscoloskeletal RX	MURA	Abnormality Detection	Binary Classification	ResNet152
[152]	MRI	BraTS, Private Datasets	Brain Tumor	Segmentation	U-Net

Table 2.2: *Overview of the task addressed by each contribution. It is worth noting that not every contribution is related to medical imaging, using EHR records as input data. Moreover, one of the contribution uses non-medical datasets in addition to medical imaging to assess the performances of the proposed approach.*

Ref.	DL Paradigm	Privacy Method	Metrics
[127]	FL, FL+DP	DP	F1
[140]	FL	HE	AUC
[141]	FL	-	AUC, Iterations, Time, Bandwidth
[142]	FL	-	Relative Bias, Error Ratio, Odd Ratio, Iterations
[143]	FL	-	MSE, ROC, AUC, Iterations, Cluster Distance
[144]	FL	-	AUC, diff. from centralized model
[145]	FL	-	AUC, Convergence to centralized sol.
[146]	FL	-	AUC, Calibration, Brier Score
[147]	FL	-	RMSE, AUC
[148]	FL	MPC	MSE
[120]	IIL, FL	-	DSC
[149]	Federated PCA	-	MSE
[119]	IIL	-	DSC, Volume Correlation
[150]	IIL	SSL	DSC + Wilcoxon, Volume Correlation
[118]	Ensembling, IIL, CCIL	-	Accuracy
[151]	Split Learning (U-shape)	-	Accuracy, DSC, AUC, Computation req, Bandwidth
[152]	FL, IIL, CCIL	-	DSC, Model Convergence

Table 2.3: *Distributed learning paradigms used by each study, along with the used privacy preserving techniques – when explicitly stated– and the metrics or measures used for assessment.*

Ref.	Data Amount Imbalance	Data Heterogeneity	Simulated Data
[127]	n.a.	n.a.	No
[140]	Balanced	Balanced (Random Sampling)	No
[141]	n.a.	Oversampled Positive Class	No
[142]	Both	n.a.	Yes
[143]	Balanced	eICU Distribution	No
[144]	Both	Disease Distribution	No
[145]	Unbalanced	Balanced	No
[146]	Unbalanced (559 vs 139 vs 196)	Stage distribution, missing data	No
[147]	Unbalanced	Label Unbalance, Incomplete data	No
[148]	Both (Simulated and Real)	Real Data Distribution	Yes
[120]	Unbalanced (BraTS), Balanced (Random Sampling)	BraTS distribution, Random	Yes
[149]	Unbalanced	Label Differences	No
[119]	Balanced	Different Acquisition procedure, Lesion Size	No
[150]	Unbalanced (18 vs 27)	Lesion Size	No
[118]	Balanced (Simulated)	Resolution	No
	Unbalanced (one institution holds fewer data)		
[151]	Balanced	IID	No
		Non-IID DRG (Simulated)	No
		BraTS modalities (T2, FLAIR)	No
[152]	Unbalanced (BraTS)	BraTS distribution	No

Table 2.4: *Overview of the dataset composition for every contribution. The column "Data Amount Distribution" reports whether the data amount in each institution is balanced or not –* simulated *indicates that the balance has been modified by the authors –, while "Data Heterogeneity" refers to differences in data distribution.*

Ref.	Paradigm Comparison	Independent Variables	N. of Nodes
[127]	CL vs FL vs FL+DP	Privacy Level, Classifier	10
[140]	CL vs FL vs SSL	Uni-Hash vs Multi-Hash	3
[141]	FL	N. of Nodes, Network Topology, Learning Methods	5, 10
[142]	FL vs IL	Data amount per client, N. of Clients	9
[143]	CL vs FL vs CBFL	N. of Nodes,	5, 10, 15, 50
[144]	CL vs FL	% of missing data, Number of patients, Number of clients, Network Structure	5
[145]	CL vs FL	-	5
[146]	CL vs FL	-	3
[147]	-	-	8
[148]	CL vs FL	N. of Samples, N. of Features, N. of Clients	3 to 12
[120]	CL vs IIL vs FL	N. of clients	4, 8, 16, 32
[149]	-	-	10, 50, 60, 100
[119]	Single Site vs IIL	Epochs	2
[150]	Single Site vs IIL	-	2
[118]	Single Site vs CL vs IIL vs CIIL	n. of clients, epochs/institution	1 to 20
[151]	Single Site (avg.) vs Split Learning	n. of clients	1 to 50
[152]	SSL vs FL vs CIL	Iteration Order, Final Model selection	10

Table 2.5: *Comparative studies between different distributed learning paradigms reported in the contributions.*

cremental Learning, using four datasets belonging to different institutions. Two datasets have been split in training and testing, while the other two are left as a holdout. Their work shows that when the collaborative model is compared with the single-site models on the two external datasets, the collaborative model performs better than every single site model. However, when the same comparison is made on the holdout datasets belonging to the clients that trained the models, the collaborative model could not reach the performance of the single-site learner that is specific for that dataset [119].

A similar result is obtained by Remedios et al., in which the multi-site model - i.e., the model trained on both NIH and VUMC datasets - performed better than both single-site models on the NIH testing dataset. On VUMC testing data, it still performed better than the NIH single-site model, while no statistically significant differences can be observed against the VUMC single-site model.

The study performed by Chang et al. on multiple datasets confirms that IIL is capable, on average, of performing within 2.5% of the performances of the centralized model. Chang et al. The authors claim that learning parallelism is not required to achieve centrally-hosted model performances. Moreover, they find that a higher frequency of weight transfer is beneficial to the testing accuracy of the final model; a boost of up to 2% in performances is also obtainable by tweaking the weight transfer frequency, depending on the dataset. Another relevant aspect is that IIL has proven robust when label imbalance or data heterogeneity -e.g., quality of data- is introduced in an institution.

The study of Sheller et al. highlights the downsides of IIL compared to Federated Learning: since IIL needs to perform validation steps frequently, the method overhead could reach those of FL. In addition, the authors claim that IIL does not scale as well as FL when the number of clients grows [120].

Investigating Split Learning for healthcare, Poirot et al. compare the performances of the U-Shaped architecture against a non-collaborative approach [151]. Their results show that Split Learning is relatively robust with each client sample size decrease. In contrast, non-collaborative solutions tend to lose performance drastically as the total data is split between multiple nodes. The study further expands the analysis of split learning by performing experiments on data heterogeneity through the application of Local Adapters.

In their study, Sheller et al. show that Federated Learning can achieve 99% of the model performances, even when data is unbalanced, or the dataset is spread among a high number of centers - e.g., 32 institutions,

holding six subjects each- [120]. Sheller et al., compared FL to (C)IIL, confirming that FL can achieve performances comparable to data sharing. Moreover, their results suggest that Federated Learning is more stable during training compared to IIL, and the latter tends to heavily bias the final model toward the last shard of data upon it has been trained [152]. The authors advocate FL as a more principled way to perform multi-institutional collaborative learning by analyzing the performances of IIL during the whole training process and highlighting the dependency of the performances on both the ordering of institutions and the final model selection strategy.

2.6 Summary

This chapter analyzed the most significant contributions in Distributed Machine Learning applied to the Healthcare setting. While the presented material is by no means exhaustive, we think it can be a good starting point for considering the current state of the research. In particular, the most prominent recent solutions when addressing imaging data or deep learning techniques are Incremental Learning and Federated Learning. This fact can be proved by the high number of available publications, especially for the Federated Learning case. Although less represented, we think that Split Learning could be a valid approach in the healthcare scenario, especially for developing personalized models that fit the needs of diverse institutions. Moreover, a comprehensive comparison with other distributed learning paradigms, such as Federated Learning, can help better understand the strengths and weaknesses of the two approaches. We investigate this issue in chapter 5.

CHAPTER 3

Datasets and Models

In the last chapter, we proposed an overview of the most used Distributed Learning paradigm. This chapter presents details on our applicative context, such as a brief overview of medical imaging (Section 3.1), the datasets used in our experiments, along with the most relevant works associated with each task (Section 3.3), and an overview of the architectures and models we used in our experiments. Lastly, we propose (Section 3.4) an overview of each machine learning task's most popular assessment metrics.

3.1 Medical Imaging Techniques

This section proposes an overview of the most commonly used medical imaging techniques.

X-Ray imaging is perhaps the most well-known and well-established kind of medical imaging. X-Rays are a form of high-energy electromagnetic radiation that can penetrate human tissues and materials. The first X-Rays have been used in clinical practice since 1896, just a few months after the discovery of this kind of radiation by W. C. Röntgen [153]. Nonetheless, X-Ray imaging is still one of the most commonly used imaging methods in medicine, and it is the primary choice for detecting pathologies of the skele-

tal system – since the calcium contained in bones absorbs X-ray radiation –. They are also widely used to perform screening of conditions involving the thoracic cavity, such as pneumonia, and can be backed up with other imaging.

X-Rays are also used in Computed Tomography (CT), which consists of a technique that involves acquiring multiple parallel scans of an anatomical region – using X-Rays or other techniques – to reconstruct a three-dimensional view of the area of interest using a *reconstruction algorithm*.

Since X-Rays are a source of risk for cancer development citeriskradiology, precautions have to be taken to ensure the safety of both patient and technologists. Moreover, they are not always adequate to diagnose certain anatomical regions, such as the brain and soft tissues. Depending on the anatomical region and the condition to investigate, a possible alternative is Magnetic Resonance Imaging (MRI). Magnetic Resonance has been introduced in the medical practice since the early seventies [154]. Instead of using X-Rays or ionizing radiations – such as the Proton Emission Tomography (PET) –, MRI uses strong magnetic fields and radio waves to generate images of the anatomical region of interest, which is generally safe.

Although many different types of imaging exist in both radiology and nuclear medicine fields, we only detail X-Ray and Magnetic Resonance imaging in the next sections, as they are the ones we have focused on in our study. The detailed physics behind X-Ray and MR Imaging is beyond our work's scope; however, in this chapter, we present the principal considerations pertinent to understanding our study's setting.

3.1.1 X-Ray Imaging

The imaging methods presented in this chapter use electromagnetic waves consisting of photons. High energy electromagnetic waves, such X-Rays are *ionzing waves* due to their ability to ionize - i.e., release an electron from an atom –. Ionizing photons can interact with matter in different ways. For example, when the interaction produces photons that deviate from the direction of the X-Ray, we have *scattering* phenomena, and depending on the setting, their effect cannot be ignored – e.g., in mammography–. A detector then captures the X-Rays; in digital radiography, it is traditionally a screen that contains phosphor, which absorbs the X-Rays that are later extracted pixel-wise using a laser beam and captured using an optical array. Resolution of X-Ray images are affected by several factors, such as the patient – thicker bodies produce more scattering, which can be attenuated by a *collimator grid* placed before the detecting surface –, the properties of the

fluorescent screen, and the sampling step at the end of the imaging chain. The image contrast depends on the detector, the attenuation coefficient, and the tissues and their thickness. In digital radiography, it can be enhanced by using a suitable mathematical transformation. The dominant kind of noise in this kind of imaging is due to the *quantum noise* of X-Rays. As the photon-detecting process can be modeled as a Poisson process, the noise amplitude is proportional to the square root of the signal amplitude, limiting the reduction of X-Rays doses without introducing noise. Other noises are due to the imaging processing, such as the conversion from photons to electrons. Lastly, the primary artifacts are due to scratches in the detector, dead pixels, and dishomogeneous X-Rays beams. However, X-Rays are generally less affected by artifacts than other kinds of imaging [155].

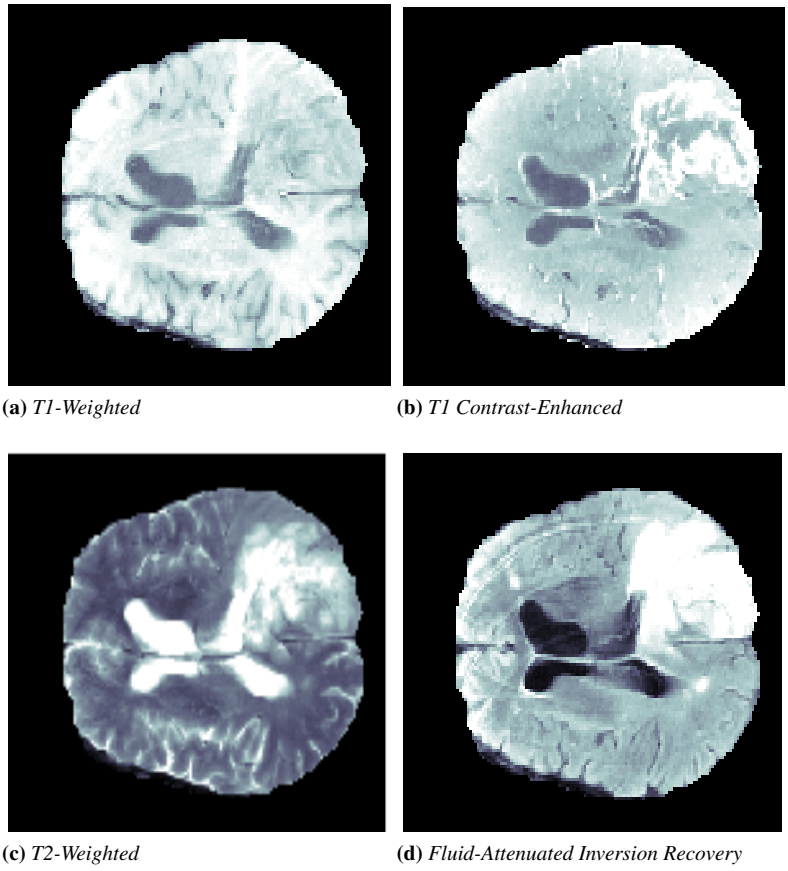
3.1.2 Magnetic Resonance Imaging (MRI)

Magnetic Resonance Imaging measures the magnetic properties of the tissues instead of capturing an image using photons. An in-depth discussion of the physics related to MRI is behind the scopes of this study. From a high-level perspective, MRI in clinical practice focuses on visualization on hydrogen-containing tissues; this is accomplished by measuring the *magnetization* of each voxel by perturbing the dynamic equilibrium of the *spins* contained in such voxel employing a sequence of radiofrequency pulses. Depending on the settings of the RF pulses and the gradient of the magnetic field used, different *MRI sequences* can be acquired. The possibility to acquire different sequences – or *modalities* – for the same images opens a set of possible applications and advantages. In our experiments, we use four of such modalities that are typically acquired for brain images:

- **T1 Weighted (T1)**: This sequence (Fig. 3.1a) shows voxels with higher water content as a low-level signal, appearing darker in the resulting image. Conversely, fat corresponds to a higher signal, resulting in brighter voxels.

- **T1 Contrast-Enhanced (T1-ce)**: The T1 sequence is sometimes *enhanced* using a *gadolinium* contrast agent, which allows visualizing blood vessels and brain tumors as brighter than the surrounding tissues. (Fig. 3.1b)

- **T2 Weighted (T2)**: This sequence is commonly acquired together with the T1 and shows a higher signal indicating more water content and a lower signal for fat. [156] (Fig. 3.1c)



(a) *T1-Weighted* **(b)** *T1 Contrast-Enhanced*

(c) *T2-Weighted* **(d)** *Fluid-Attenuated Inversion Recovery*

Figure 3.1: *Four different MRI Modalities of the same brain affected from glioma. Source: BraTS Dataset [31]*

- **Fluid-Attenuated Inversion Recovery (FLAIR)**: This kind of sequence nullifies liquid content such as the cerebrospinal fluid, which appears dark. The rest of the image is similar to T2 weighted images. [157] (Fig. 3.1d)

It is worth noting that these are not the only MRI sequences possible. Most notably, for the brain district, *Susceptibility-Weighted* MRI can detect small hemorrhages [158], *perfusion weighted* MRI can be used to measure blood flow [159], and functional MRI (fMRI) can be used to localize brain activity when performing a particular task [160].

The contrast in an MRI depends on the tissue, and the parameters used when defining the sequence. Resolution is typically between 2mm and 1mm, depending on the magnetic field strength that the imaging system

can develop. The primary source of noise is thermal, and it is affected by the temperature of both the patient and the MRI system. Artifacts found in MRI are mainly due to technical issues, such *dephasing* due to the RF field not being homogeneous, numerical approximations, insufficient shielding of the room, or interactions with other equipment. [155]

3.2 Datasets

3.2.1 Microsoft Common Object in Context (COCO) Dataset

The Common Object in Context (COCO) [161] is a Dataset for instance segmentation made available by Microsoft in 2015. Instance segmentation is a different task from Semantic Segmentation: while the latter focus only on classifying each pixel of an image according to a label that identifies the kind of object - e.g., identifying a Dog in an image-, the former takes a step further and also allows to identify different objects of the same kind in the same image -e.g., telling which pixels belongs to two different dogs depicted in the same image-. COCO contains a total of 2.5 million labeled object instances in 328k images. The dataset contains 91 common object categories; however, we used only a smaller subset in our work. While COCO is not a healthcare-related dataset, it can easily benchmark deep learning models for image classification or segmentation. This can be beneficial for training models with less effort, as most medical image datasets are typically much smaller and with more skewed distributions than ordinary object datasets. Another advantage is that visual inspection of healthcare data requires expert knowledge. The author of COCO chose instead objects that *"would be easily recognizable by a 4-year-old"*, allowing a much more straightforward interpretation of segmentation results.

3.2.2 Brain Tumor Segmentation (BraTS) Dataset

The MICCAI Brain Tumor Segmentation (BraTS) [162, 31] challenge focuses on evaluating methods for segmenting brain tumors, particularly gliomas, the most common primary brain malignancies.
The BraTS dataset comprises un-normalized, isotropic MRI volumes of resolution $240x240x155$ and four contrast modalities: T1, T1ce (contrast-enhanced), T2, and FLAIR.

Each MRI modality can capture different physical properties of the tissues; for this reason, each of them generally includes a different kind of information.

Each voxel is annotated with one among five possible labels representing the tumor sub-regions:

- *Enhancing/Active Tumor* (ET) Represents the area of the tumor that is absorbing the Gadolinium-based contrast agent. For this reason, this area is generally more intense on a T1c image concerning the T1 counterpart or the rest of the healthy white matter.
- *Non-Enhancing Tumor* (NET) represents the solid area of the tumor that is not absorbing the contrast agent. It is typically more intense in T1 than in T1c.
- *Necrosis* (NCR): Corresponds to the fluid-filled tissues of the tumor. It is typically more intense in the T1 modality than in T1c.
- *Edema* (ED) Represent the peritumoral edema, and it typically corresponds to a hyper-intense signal in the FLAIR modality.
- *Everything Else* represents the rest of the image, being both healthy tissues or voxels outside the region of interest.

The volumes are collected across different institutions with a diverse contribution in terms of the number of samples. Over the years, multiple versions of the dataset have been released. In the next paragraph we describe only the versions that we used in our work, i.e. BraTS 2015 [162] and BraTS 2019 [31].

The BraTS 2015 and 2019 are not disjoint, as the 2019 version improves the older dataset. Here we report the most relevant differences between the two datasets: In terms of dataset size, the BraTS 2015 dataset consists of 220 high-grade gliomas (HGG) and 54 low-grade gliomas (LGG) MRIs, while BraTS 2019 consists of a total of 335 MRI volumes (259 HGG, 76 LGG);

1. Both datasets contain 30 human-annotated MRIs from a previous dataset, BraTS 2013;
2. The remaining volumes in BraTS 2015 are a mixture of pre and post-operative scans that have been labeled by an ensemble of algorithms and later evaluated by a team of human experts. In BraTS 2019, all the post-operative scans have been discarded, and all the volumes have been manually re-labeled;
3. Additional data from different institutions have been included in BraTS 2019;

4. In BraTS 2019, the *Non-Enhancing Tumor* (NET) label has been merged with *Necrosis* (NCR) due to a bias that exists in the evaluation of that area.

In our experiments, we will generally use *"BraTS dataset"* to refer to the 2019 version, unless explicitly stated.

The challenge also defines some sub-regions for evaluating the results by composing the various labels:

1. The *Enhancing Tumor*, corresponding to the label with the same name
2. The *Tumor Core* (TC) entails the Enhancing Tumor, the necrotic (NCR), and non-enhancing (NET) parts of the tumor. This region is the region that is typically resected as it represents the bulk of the tumor.
3. The *Whole Tumor* (WT) includes the Tumor Core and the Edema.

In our work, unless otherwise specified, we report our results on the Whole Tumor (WT) region.

3.2.3 Stanford Chest X-Ray (CheXpert) Dataset

Chest X-Ray Diagnosis is a task that can be addressed using CNN classifiers. A large dataset labeled with good quality standards is necessary to this extent. In our experiments, we focused on the CheXpert dataset [163], which is composed of 223316 Chest X-Rays of 65240 patients, collected from the Stanford Hospital from October 2002 to July 2017. The dataset is provided in two different image formats: the high-quality format is 16-bit PNG, and the low-quality format is 8-bit PNG.

Each image is annotated with a vector of 14 labels, corresponding to significant findings in a Chest X-Ray. The labels have been extracted from text radiology reports using an automatic rule-based labeler.

In particular, the labeling process consisted of three different phases:

1. the *Impression* section – that generally summarizes the key finding of the exam – of each report is analyzed, and a list of mentions is extracted, by matching a list of phrases designed by multiple expert radiologists;
2. each mention is assigned to a label according to a level of confidence between *positive*, *negative* and *uncertain*;
3. each image is encoded as a vector that includes one element for each label: positive labels are encoded as 1, negative labels are encoded as 0, uncertain labels are encoded as u.

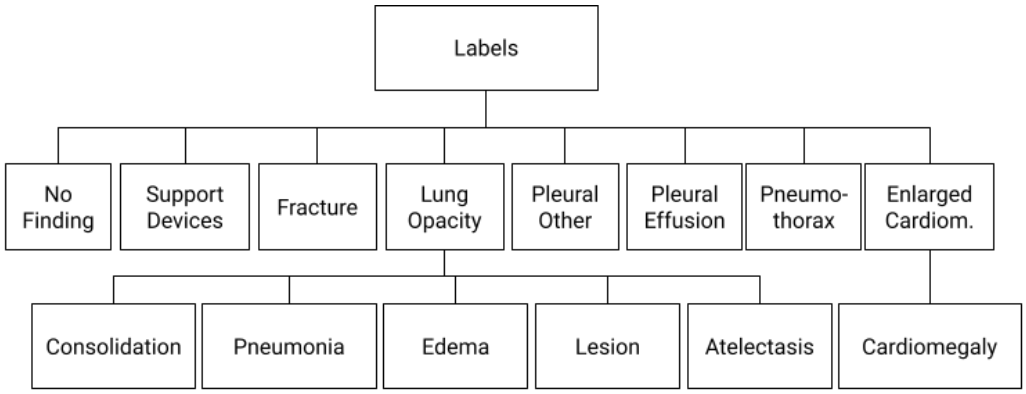


Figure 3.2: *Hierarchical structure of the findings. This is a slightly simplified version of the hierarchy described in [164].*

CheXpert dataset includes two kinds of images: frontal and lateral X-Rays. Lateral images are available only for some patients, generally when the diagnosis is uncertain. For this reason, the amount of frontal images is much higher than those of the lateral ones. In addition to the training set, the authors also provide a test set of 200 images – annotated by human experts – that can be used to assess the performances of the machine learning approaches.

Table 3.1 shows the data distribution over the 14 labels included in the dataset. The 14 findings expressed by the labels represent medical conditions that are not independent, but rather they form a hierarchy as shown in Figure 3.2. The hierarchy models correlations between the labels, e.g., Pneumonia is correlated with the observation of Lung Opacity, while Cardiomegaly can be correlated with Enlarged Cardiomediastinum.

In order to be successful, a machine learning approach applied to the CheXpert dataset would need to deal both with the uncertainty of the labeling process and the dependency among the labels, which could be exploited to improve the performances.

Along with the publication of the CheXpert dataset, Irvin et al. [164] also proposed their own solution based on a 121-layer DenseNet trained with different approaches to deal with the uncertainty present in the CheXpert dataset labels. Their model was able to achieve performance similar or better than expert radiologists on the classification of 5 thoracic diseases, selected as the most representative of the dataset. Finally, in a very recent work, Pham et al.[165] trained several state-of-the-art CNN on the CheXpert dataset, showing the benefits of exploiting the conditional dependen-

Pathology	Positive (%)	Uncertain (%)	Negative (%)
No Finding	16974 (8.89)	0 (0.0)	174053 (91.11)
Enlarged Card.	30990 (16.22)	10017 (5.24)	150020 (78.53)
Cardiomegaly	23385 (12.24)	549 (0.29)	167093 (87.47)
Lung Opacity	137558 (72.01)	2522 (1.32)	50947 (26.67)
Lung Lesion	7040 (3.69)	841 (0.44)	183146 (95.87)
Edema	49675 (26.0)	9450 (4.95)	131902 (69.05)
Consolidation	16870 (8.83)	19584 (10.25)	154573 (80.92)
Pneumonia	4675 (2.45)	2984 (1.56)	183368 (95.99)
Atelectasis	29720 (15.56)	25967 (13.59)	135340 (70.85)
Pneumothorax	17693 (9.26)	2708 (1.42)	170626 (89.32)
Pleural Effusion	76899 (40.26)	9578 (5.01)	104550 (54.73)
Pleural Other	2505 (1.31)	1812 (0.95)	186710 (97.74)
Fracture	7436 (3.89)	499 (0.26)	183092 (95.85)
Support Devices	107170 (56.1)	915 (0.48)	82942 (43.42)

Table 3.1: *Number of Positive, Uncertain and Negative samples for each finding.*

cies among the labels in training as well as of employing an ensemble of classifiers with different architectures instead of a single one.

It is worth mentioning that the CheXpert dataset is not the only Chest X-Ray dataset to perform automated diagnosis. For example, Rajpurkar et al. [166] trained a DenseNet-121 [167] model on the ChestX-ray14 dataset [168]; their model – dubbed *CheXNet*–, achieved state-of-the-art performances on the classification of the 14 major thoracic diseases and outperformed expert radiologists on the detection of pneumonia. In a later work [169], Rajpurkar et al. introduced *CheXNeXt*, improving the performance of *CheXNet* and achieving a performance similar to expert radiologist on 10 thoracic conditions. Notable works that focus on the ChestX-ray14 dataset are also the work of Kumar et al. [170], who introduced cascaded CNNs that can diagnose all the 14 thoracic diseases better than the baseline, and the work of Lu et al. [171], who applied an evolutionary algorithm to search for the optimal CNN architecture to solve the classification task. A different approach was followed by Ye et al. [172] that introduced *Probabilistic-CAM* (PCAM), an extension of CAM [45], to perform the localization of thoracic diseases on the ChestX-ray14 dataset in a semi-supervised fashion. At the same time, the localization model trained can be successfully applied also to solve the image classification problem with a performance similar or better than some of the previous approaches in the literature, such as *CheXNet* [168]. Lastly, working with frontal and

lateral images, Rubin et al. [173] introduced *DualNet*, consisting of two CNNs jointly trained on frontal and lateral chest radiographs, included in the very large MIMIC-CXR dataset [174]. Their results show that *DualNet* outperforms state-of-the-art classifiers trained separately on a single type of image (i.e., either frontal or lateral).

3.3 Models

This section describes all the machine learning models and architectures that we used in our work. Although a significant number of new models and architectures are proposed each year in the machine learning field, we decided to use the most popular ones for our experiments. This is because this work aims to be a compendium of possible techniques to exploit when addressing issues specific to collaborative learning for healthcare or medical imaging. By choosing popular architectures, we promote the reproducibility of results, while more complex models could be added afterward to improve the results further. Nonetheless, we also explored novel techniques such as Adversarial Networks for Segmentation, as we show in chapter 6, as we envision that they represent an interesting approach to the problem that could bring promising results as the research progresses further. The following section organizes the models according to the learning task they address. This section can thus be used as a reference for the remainder of this manuscript.

3.3.1 CNN Architectures for Image Classification

In our studies, we often resorted to well-known CNN architectures to exploit the achievements already part of state-of-the-art. This section summarizes the neural network for classification that is part of the Keras applications library[175]. These neural networks have been pre-trained for the ImageNet dataset. However, we specify whether we used pre-trained or random weights in each experiment that uses such architectures. Table 3.2 shows for each architecture the number of weights, the number of layers, the size in MB of the weights, and the corresponding article.

3.3.2 CNN Architectures for Image Segmentation

In 2015, Ronneberger et al. proposed U-Net [179], a Convolutional Neural Network for biomedical image segmentation. The name is due to the visual representation of the network, which is composed of two paths: the first is a *contracting* path, which is composed of repeated blocks of two

Name	#Parameters	Depth	Size	Year
DenseNet121	7M	121	33MB	2016 [167]
DenseNet169	12,5M	169	57MB	2016 [167]
DenseNet201	18M	201	80MB	2016 [167]
InceptionResNetV2	54M	572	215MB	2016 [176]
Xception	21M	126	88MB	2016 [177]
VGG16	15M	23	528MB	2014 [178]
VGG19	20M	26	549MB	2014 [178]

Table 3.2: *List of pre-trained CNN used in our work*

3x3 convolutions, a ReLU [86] activation function and a 2x2 *Max Pooling* operation to reduce dimensionality. After each block, the number of feature channels is doubled. The contracting path performs feature extraction from the images, and it is followed by a symmetrical *expansive* path. The blocks in the expansive path use 2x2 *transposed convolution*, which halves the number of feature channels. After each convolution, the features are concatenated with the features coming from the corresponding *contracting* block – this kind of layer is often informally referred to as *skip layer*. After the concatenation, two 3x3 convolutions and a ReLU activation function are applied. The final layer comprises a 1x1 convolution operation, which maps each feature vector in the target number of classes. U-Net has in total 23 convolutional layers. The peculiarity of this network is that the *skip connections* allows lower level features to "skip" the innermost layers of the network and reach the expanding path at the right level. As confirmed by our findings in Chapter 2, and the BraTS Challenge results [162, 180], U-Net proved to be a successful approach for segmentation, often establishing as a valuable building block for more complex architectures.

Adversarial Models for Segmentation

As discussed in Chapter 2, GANs can be modified to perform segmentation by altering the configuration of their input and outputs. To this extent, Luc et al.[181] proposed a method in which a segmentation network is trained to perform pixel-wise classifications on images. In contrast, an adversarial network (called *discriminator* or *critic*) is trained to discriminate segmentations coming from the segmentation network and the ground truth. This approach has been tested on the *PASCAL VOC 2012* [182] and on the *Stanford Background* [183] datasets, and the results show an improvement in segmentation performances when an adversarial loss is used. In the medical imaging domain, multiple works applied adversarial methods to segment MRI, CT, PET, and other domain-specific image formats [97]. In particular,

two previous works applied adversarial learning to Brain MRI. Moeskops et al. [184] investigated the effectiveness of adversarial training and dilated convolution. Xue et al. [185] proposed an Adversarial Network with a Multi-Scale loss, called *SegAN*, achieving better performances compared to the state-of-the-art methods for brain tumor segmentation [186, 187]. In Chapter 6, we will present an extension of the SegAN architecture in the context of investigating the transfer learning capabilities of Adversarial Networks.

3.3.3 CNN Architectures for Image Translation

As we discussed, image generation became popular thanks to the introduction of GANs. Following this approach, Isola et. al. introduced *pix2pix* [63] in 2016. The idea behind pix2pix is to exploit conditional GANs [96] to perform translation between multiple images. As with (c)GANs, the network is composed of a generator and a discriminator network. The two networks resemble the ones used in DCGAN [95], with added *skip connections* for the generator – similarly to U-Net– and a PatchGAN [188] as the discriminator, to only penalize structures of a specific scale.

The loss function is composed of two contributions, a cGAN loss:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[logD(x, y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x, z)))] \quad (3.1)$$

where x are the observed images, y are the output images and z is a random noise vector, and a L_1 Loss:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||G(x, z)||_1] \quad (3.2)$$

that forces the generator not only to fool the discriminator but also to generate samples that are similar to the ground truth. The author claims that, although this approach has been already used with an $\mathcal{L}_2$ distance, a $\mathcal{L}_1$ norm is beneficial as it induces less blur in the resulting images.
Another popular architecture for image translation is CycleGAN [65], that is characterized by having two generators and two discriminators, which are related using a *cycle consistency loss*. This kind of approach allows to generate samples of a source or target domain without disposing of matched data pairs - e.g., having both modalities for every image.

3.3.4 Classifiers based on Trees

While CNN proved to be very successful in classifying image data, it may be helpful to combine their results with other kinds of classifiers, as we

show in chapter 6. For this reason, we also considered two well-known classifiers that are based on trees.

The first approach is Random Forests [189] (RF), which is an ensemble learning method that can be used for classification. It is based on the technique of *bagging* [190] applied to *decision trees*, in order to reduce the variance of the model by averaging multiple trees trained on different subsets of data. Additionally to bagging, a random selection of features is performed every time a tree is generated to decrease the correlation between different trees and promote a gain of accuracy.
The second approach is called eXtreme Gradient Boosting [191] (XGBoost). *Gradient Boosting* is an algorithm that incrementally combines multiple *weak learners* into a stronger one. In particular, every newly added classifier is trained, focusing on the misclassified samples of the previous. In XGBoost, Gradient Boosting is applied using Decision Trees as classifiers.

3.4 Assessment Metrics

3.4.1 Metrics for Classification and Segmentation

For classification tasks, assessment metrics are usually based on the *Confusion Matrix* [192]. Given the output of a network model, the output is converted into a label, usually using a threshold. The predicted class is then compared with the actual label of the model and labeled as one of *True Positive (TP)* indicating a correct classification, *False Positive (FP)* indicating a false alarm, *True Negative (TN)* correct rejection or *False Negative*, indicating a miss.
Two commonly used metrics derive directly from these values are:

(i) the *precision* - or *Positive Predictive Value* –:

$$PPV = \frac{TP}{TP + FP} \tag{3.3}$$

measures how many samples, classified as positive, are effectively positive in the ground truth;

(ii) the *sensitivity* – also, dubbed *recall* or *True Positive Rate* –

$$TPR = \frac{TP}{TP + FN} \tag{3.4}$$

measures how many samples, that are positive in the ground truth, are correctly identified as such;

(iii) the *specificity* – or *False Positive Rate*–

$$FPR = \frac{FP}{FP + TN} \tag{3.5}$$

measures how many sampes, that are negative in the ground truth, are correctly identified as such;

Accuracy

The accuracy is defined as:

$$ACC = \frac{(TP + TN)}{(TP + FN + FP + TN)} \tag{3.6}$$

It measures how many samples are correctly classified by the model. However, it is generally not considered a good metric to optimize for a Deep Learning classifier: if the dataset is highly unbalanced, as often applies in medical datasets or in segmentation, a model that only predicts the majority class –e.g., the negative class– would produce misleadingly high values of accuracy.

F1 Score and Dice Score

The F1 score [193] is another popular metric for evaluating a classifier. It is defined as the harmonic mean between precision and sensitivity:

$$F_1 = 2 * \frac{1}{\frac{1}{PPV} + \frac{1}{TPR}} \tag{3.7}$$

In computer vision, the F1-Score is often referred to as Dice Score, although the latter was intended to be applied to discrete data [194, 195]. When applied to binary data, the Dice Score is computed using the formulation:

$$DSC = \frac{2 * TP}{2 * TP + FP + FN} \tag{3.8}$$

In a segmentation task, the Dice Score can be seen as a metric of *overlap* between the ground truth segmentation and the one generated by the segmentation model. A value of 1 indicates complete overlap, while 0 indicates no overlap.

Area Under the Receiving Operating Characteristic

The Area Under the (*Receiving Operating Characteristic*) is one of the most widely used metrics for evaluating classifiers. The ROC curve is obtained by plotting the True Positive Rate (TPR) against the classifier's False Positive Rate (FPR) at various threshold values. As a reference, an AUROC value of 0.5 means no discriminative power, while – in the medical field – a value between 0.7 and 0.8 is considered acceptable, a value between 0.8 and 0.9 is considered excellent, and values larger than 0.9 are considered outstanding [196].

Matthew's Correlation Coefficient (MCC)

The Matthew's Correlation Coefficient (MCC) [197] is a correlation coefficient equivalent to the Pearson's Phi Coefficient [198]. It can be calculated using the formula:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.9}$$

Since this metric is a correlation coefficient between the predictions and the ground truth, it must be interpreted accordingly. A value of +1 indicates a strong correlation with the ground truth, a value of 0 indicates that the classifier is no better than a random guess, while -1 indicates anti-correlation between the prediction and ground truth.

3.4.2 Metrics for Regression and Image Generation

The regression and image generation models output one or more scalar values instead of categorical ones; a set of metrics defined on this kind of data is thus required. In this section, we indicate as y_i the predictions of the considered model and as $\hat{y}_i$ the values observed in the ground truth, with $i \in 1, ..., N$ representing the index of observations and predictions. Although a large number of other metrics are available, we only report the most used ones, according to our findings of chapter 2.

Mean Absolute Error (MAE)

The most straightforward metric is the Mean Absolute Error (MAE):

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{3.10}$$

The MAE is simple and efficient to compute, and values toward 0 indicate better model performances. However, it has the drawback of being heavily dependant on the error magnitude, rendering it challenging to use when the error is significant.

Mean Squared Error (MSE)

Another popular approach is the Mean Squared Error (MSE):

$$MSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2|}{N} \tag{3.11}$$

Like MAE, MSE tends to return large values when the error is high. Compared to MAE, it penalizes larger errors; for this reason, it is often used as part of loss functions.

When dealing with images, it is a common practice to use MSE to assess the similarity between two images:

$$MSE(\hat{y}, y) = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} (\hat{y}_{i,j} - y_{i,j})^2, \tag{3.12}$$

where W and H are the width and the height of the image in pixels. This approach, however, has the drawbacks of not accurately reflecting the perceptual similarity between the images – i.e., when the two images are inspected by a human observer. For this reason, we also introduce two measures to deal with this problem.

Peak Signal-to-Noise Ratio (PSNR)

The PSNR [199] is commonly used to evaluate images corrupted by noise. It is calculated as:

$$PSNR(y, \hat{y}) = 10 \log_{10} \frac{(\max(y_{i,j}))^2}{MSE(y, \hat{y})} \tag{3.13}$$

where MSE is the mean squared error between the two images. The higher the value of PSNR, the better the quality of the generated image.

Structural Similarity (SSIM)

$$SSIM(y, \hat{y}) = \frac{(2\mu(\hat{y})) + \mu(y) + c_1)(2\sigma(y, \hat{y}) + c_2)}{(\mu^2(\hat{y}) + \mu^2(y) + c_1)(\sigma^2(\hat{y}) + \sigma^2(y) + c_2)}, \tag{3.14}$$

where μ is the average of pixel values, σ^2 is the variance of pixel values, and $\sigma(y, \hat{y})$ is the covariance of y and $\hat{y}$ pixel values, and c_i are regularization constants for luminance, constants and structural terms. [200].

CHAPTER 4

Ensemble Learning

The first class of distributed learning paradigms we analyze is Ensemble Learning. Ensemble Learning algorithms have the advantage of being among the simplest to implement; moreover, they only rely on the outputs of previously trained models, making them a feasible solution to adopt when a set of learners is already available at each node.

In this chapter, we investigate two different aspects related to Ensemble Learning. In the first setting, we assume to have a set of different *segmantation models*, trained on heterogeneous data. To this extent, we first introduce a set of ensembling strategies. Some of the proposed strategies are general enough to be applied to classification tasks, while others – i.e., those based on local features of the predictions – are specific for segmentation. To validate our approach, we first run an experiment using a dataset of *simulated predictions*; this allows us to study the performances of the ensembling methods without the limitations of data availability. Then, we proceed by training different sets of machine learning models. To overcome the scarcity of data, we use a non-medical dataset – i.e., the COCO dataset – to assess the performances of the proposed method when using real machine learning models. Lastly, we investigate our approach to medical data, precisely the Brain Tumor Segmentation task. In this context, our

aim is not to build the best possible segmentation model but to evaluate under what circumstances an ensembling method is preferable to another. The second aspect we analyze is related to a setting where heterogeneity is not on the data but in the model architectures. To this extent, we use the problem of Chest X-Ray Diagnosis as a case study. In particular, we train seven different CNN architectures on the CheXpert dataset, and we apply different ensembling techniques to improve their performances while at the same time obtaining a single model from all the seven architectures. The results of this experiment will constitute the starting point of the analysis for the following Chapters when the same task is addressed to study different aspects of collaborative learning.

4.1 Ensembling Methods for Image Segmentation

In this section, we propose our Ensembling Methods for Image Segmentation. While ensembling methods are a popular technique for aggregating knowledge in a regression or classification context, we envision adapting the same techniques to image segmentation. This kind of data opens more possibilities in terms of designing new solutions that exploit the inductive bias of medical images - i.e., the probability that pixels that are spatially close to each other shares common features are higher than the pixels that are far apart-. Working on this, we also designed some aggregation methods that take into account the spatial relation of the pixels. In this section, we first formally introduce the concept of Collaborative Segmentation, then we propose different aggregation strategies. Lastly, we present our experiment design and results on three different datasets.

4.1.1 Collaborative Segmentation

In this section, we address the problem of image segmentation in a collaborative context. We define *collaborative image segmentation* as the problem of, given a set of image segmentation models, providing a single prediction that takes into account the outputs of the single contributions. In this context, an image segmentation model can be any statistical predictive model that, given an $\mathrm{H} \times \mathrm{W} \times \mathrm{C}$ input image x, produces for each pixel a probability distribution over a set of *classes*, or labels $\mathcal{L}$.

More formally, we consider the setting in which there are N agents $a_i \in \mathcal{A}; i \in 0, ..., N$, each one equipped with a *segmentation model* that is trained on a different subset of data. The same input image is proposed to every agent, which provides its model prediction as a *solution proposal* for that input image. Each proposal $P_a \in \mathcal{P}, a \in \mathcal{A}$ is a matrix of pixels $p_{a,x,y,l} \in$

$[0, 1]$ where $a \in \mathcal{A}$ is an agent, $x \in \{0, ..., W\}$ and $y \in \{0, ..., H\}$ are the coordinates of the corresponding input pixel and $l \in \mathcal{L}$ is an output label.

We considered the class of problems in which the predicted labels are mutually exclusive. In other words:

$$\sum_{l \in \mathcal{L}} p(a, x, y, l) = 1 \forall a \in \mathcal{A}, x \in \{0, ..., W\}, y \in \{0, ..., H\} \tag{4.1}$$

To benchmark our proposed methods, we also assume that for every input image x, a *ground truth* $t_{x,y,l} \in \mathcal{T}$ is available. Every ground truth matrix has the same format and properties of a single agent proposal P_a.

For simplicity, we only considered *binary segmentation*, so in our experiments, the number of labels is always 2 -either the *target label*, or *everything else*- even when we worked with a dataset containing multiple labels. This has been accomplished by considering each label as a new binary segmentation task.

4.1.2 Aggregation Methods

To aggregate the knowledge for the different predictions, we propose different strategies. Every strategy has the objective to combine the N agent proposals P_a to obtain a final proposal P_f that represents the collective decision on a given input image. We omit the pixel coordinate (x, y) when the formula refers to the same spatial location to keep the notation simple. Each strategy produces a matrix of floating numbers that satisfies eq.4.1. We adopt the decision strategy to choose the label with the maximum probability after the aggregation. If there are ties in selecting the final labels (i.e., two or more labels obtain the same maximum score), we perform random sampling among them.

We designed two kinds of aggregation methods: (1) *simple methods* combines the numerical output of each agent without extracting or modeling any additional quantities. (2) *confidence-based methods*, which model a measure of uncertainty in the prediction of an agent prior to aggregating the proposals.

4.1.3 Simple Methods

Maximum Proposal The *maximum proposal strategy* select, for each pixel, the label that has been assigned the highest probability among all the predictions.

$$p_{max}(l) = \frac{\max_a p(a, l)}{\sum_{l \in \mathcal{L}} \max_a p(a, l)} \tag{4.2}$$

Mean Proposal The *mean proposal strategy* selects, for each pixel, the mean proposal from all the agents

$$p_{mean}(l) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} p(a, l) \tag{4.3}$$

Majority Voting The *majority voting strategy* selects, for each pixel, the class that has been voted by the majority of agents. We first calculate the votes:

$$v(l) = |\{a \in \mathcal{A} : \arg\max_{i \in \mathcal{L}} p(a, i) = l\}| \tag{4.4}$$

that is a function that express, for each pixel, how many agents proposed the label l with an higher probability than the others. Then, the *majority voting* is calculated as:

$$p_{mv}(l) = \begin{cases} 1 & \text{if } l = \arg\max_{i \in \mathcal{L}} v(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

4.1.4 Confidence-Based Methods

Machine Learning algorithms could include a measure of uncertainty of the prediction. However, the Convolutional Neural Networks commonly used for segmentation do not provide such an output. However, to investigate how the uncertainty of a prediction can affect the collaborative output of a set of models, we developed a heuristic to model uncertainty based on the model output directly. We define the *proposal uncertainty* based on the *Information Entropy* [201] over the probabilities associated with each label:

$$\mathcal{H}(a) = -\sum_{l \in \mathcal{L}} \frac{p(a, l) \log p(a, l)}{\log |\mathcal{L}|} \tag{4.6}$$

The choice of this function to model uncertainty is based on the assumption that we might model the prediction of each classifier as a random variable that follows a Bernoulli distribution with a success probability equal to the actual output of the classifier. We can thus use the Entropy value of such a variable to measure the classifier confidence. In other words, the distribution of the label probabilities of an uncertain prediction is more uniform than the distribution of a particular prediction, which would show a value close to 1 for one label and values close to 0 for the others. Indeed, this function would have the value of 1 when all the outputs for a pixel have value $\frac{1}{|\mathcal{L}|}$ (maximum uncertainty) and 0 if all but one output are set to zero (minimum uncertainty).

Since most segmentation models account for local features to compute the prediction of a single pixel, we also investigated different methods to include the uncertainty of a sample. In particular, in order to correctly include the uncertainty measure in our aggregation methods, we defined different *confidence functions* based on the proposal uncertainty and developed three different strategies:

Pixelwise Confidence represent the simplest case, in which the confidence function for a pixel at coordinates (x, y) is:

$$c_{pw}(a, x, y) = 1 - \mathcal{H}_{a,x,y} \tag{4.7}$$

Convolutional Confidence: In this case, we apply the 2D Convolution operator $(*)$ to perform a computationally-efficient local mean in a patch of either 3x3 or 5x5 pixels.

$$c_{SxS}(a, x, y) = c_{pw}(a, x, y) * W_S \tag{4.8}$$

Where S is the patch size (either 3 or 5) and W_S is a matrix of constant weights $\frac{1}{s^2}$ of shape $SxSx|\mathcal{L}|$.

Full-Image Confidence With this method we consider the overall confidence for the full image. For each pixel we compute the confidence function as

$$c_{full}(a, x, y) = \frac{1}{HW} \sum_{i=0}^{W} \sum_{j=0}^{H} c_{pw}(a, i, j) \tag{4.9}$$

where H and W are the image dimensions.

Each Confidence-Based method is the combination of two orthogonal dimensions: (i) a suitable *Aggregation Method* that will be presented and (ii) the *confidence function* (Pixelwise, Conv. 3x3, Conv. 5x5, Full). that have been described in the previous section.

Weighted Mean With this method we perform a weighted mean of the proposals, using as weights the values provided by each confidence function:

$$p_{wm}(l) = \frac{\sum_{a \in \mathcal{A}} p(a, l) c(a)}{\sum_{a \in \mathcal{A}} c(a)} \tag{4.10}$$

with $c(\cdot)$ being one of the possible confidence functions.

4.1.5 Experiments on Synthetic Data

When comparing the different aggregation methods, it is necessary to analyze multiple scenarios, as a method could perform differently according to the considered agents and their performances on inference data. Moreover, the healthcare domain also poses additional challenges since datasets sizes are insufficient to analyze all the possible scenarios. Finally, most public datasets do not track information about the source of each sample or different modes in the data distribution. To address the problems mentioned above, we designed three different experiments. The first experiment we made is a simulation made using synthetic data. This experiment explores the impact of the different choices in some typical scenarios. In particular, we consider the level of confidence of the agent and their predictive performances. We will test our ensembling approaches on image datasets from two domains in the following sections.

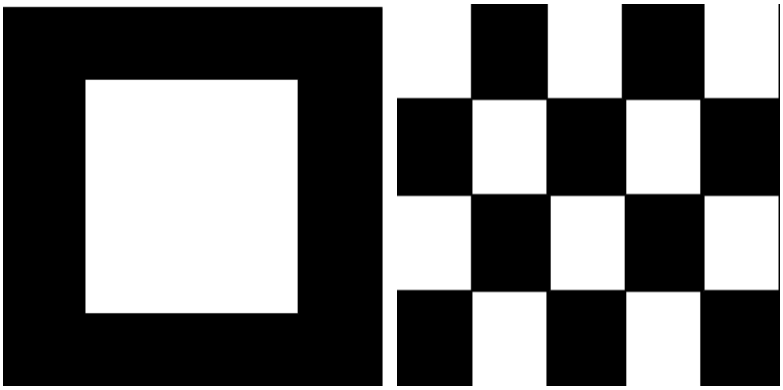
Modeling the Agents

To study the possible real-world scenarios more closely, we directly designed a synthetic dataset of predictions $p(a, x, y, l)$ to model each agent behavior concerning a specific input. Each synthetic prediction is handcrafted for every pixel of an input pattern to study a particular condition in which each agent could operate. This experiment considers the agent capabilities as a prediction model and some typical spatial configuration of the input samples. In particular, for each test case, we model:

1. The *template* for the ground truth, which is shown in fig. 4.1, that is related to the shape of the continuous area that is segmented with the same value: we considered either a single squared area, named "blob", or a checkerboard-like pattern of alternating classes.

2. Each agent *proposal*: the proposed solutions are built according to a probability matrix similar to a *confusion matrix*. This matrix models the probability P_{ij} of an agent to predict a label $\mathcal{L}_j$ when the true label is $\mathcal{L}_i$.

In a binary segmentation task, each agent is required to classify each pixel with one from two classes (e.g., C_1 and C_2). We model the probability of choosing a class with a normal distribution for each pixel, given the true class of the ground truth. The following scenarios control the mean μ of the distribution given each class, while the variance σ is the same for each possible choice.



(a) *Binary Blob* **(b)** *Binary Checkerboard*

Figure 4.1: *Ground Truth Templates used for the synthetic dataset. White areas and black areas represent two different classes – e.g., C_1 and C_2 – in the ground truth.*

Table 4.1: *Agent parameters for a Balanced Agent.*

		Predicted	
		C_1	C_2
True	C_1	$\mathcal{N}(\mu, \sigma)$	$\mathcal{N}(1-\mu, \sigma)$
	C_2	$\mathcal{N}(1-\mu, \sigma)$	$\mathcal{N}(\mu, \sigma)$

- **Balanced Agent**: A balanced agent has the same mean probability μ of choosing the *correct* label, as opposed to $1 - \mu$ of choosing the wrong one. In other words, it does not have any bias in proposing either the correct class or the other.
- **Unbalanced Agent (on class C_i)**: An unbalanced agent is an agent that has two different probabilities μ_1 and μ_2 (with $\mu_1 > \mu_2$) of choosing the correct class, when it is C_1 and C_2 respectively. In other words, an unbalanced agent is an agent that has better performances on a class with respect to the other.

Lastly, every agent proposal is affected by the variance σ. This parameter represents the noise in the input data that adversely affect the prediction of a machine learning model.

Generating the Synthetic Dataset

Once we defined the different types of agents, we generated a dataset of synthetic proposals to be aggregated with the different ensembling strategies. We ran, for each ground truth template, agent parameter and aggrega-

Table 4.2: *Parameters for a Binary Agent Unbalanced on C_1.*

		Predicted C_1	Predicted C_2
True	C_1	$\mathcal{N}(\mu_1, \sigma)$	$\mathcal{N}(1-\mu_1, \sigma)$
	C_2	$\mathcal{N}(1-\mu_2, \sigma)$	$\mathcal{N}(\mu_2, \sigma)$

tion method, the following tests:

- 1 Balanced agent vs. 2, 4, or 8 Unbalanced agents on C_1
- 1 Balanced agent vs. 2, 4, or 8 Unbalanced agents on C_2
- 1 Balanced agent vs. 2, 4, or 8 Unbalanced agents, half on C_1 and half on C_2

The values of μ_b of the balanced agent belong to the set $\{0.6, 0.75, 0.9\}$. The values of μ_1 and μ_2 belong respectively to $\{0.6, 0.75, 0.9\}$ and $\{0.4, 0.5\}$ for the agents unbalanced on C_1, while they are swapped for the agents unbalanced toward C_2. Finally, the standard deviation for every kind of agent can take the values $\{0.01, 0.05, 0.1, 0.2\}$. For each possible set of parameters, we ran 30 simulations by resampling the predictions.

In some cases, it is possible that all the agents agree on a particular class for every pixel in the image or that the area in which they do not agree on class is minimal compared to the image size. Thus, we define as *conflict area* the union of pixels in the final aggregation for which at least one agent voted for a different class from the others. For this reason, we analyze our results only on those images for which the conflict area is greater than zero. Considering the areas in which every agent agrees on the solution might make it more difficult to compare the different methods if the size of the agreement is much larger than the conflict area.

Results

The first results we present are on the synthetic dataset of proposals. To ease the analysis of the results, we use the following independent variables: the μ_b of the balanced agent, the number of unbalanced agents, the noise σ introduced in the prediction, and the ground truth pattern. The results are computed by averaging all possible combinations of the other parameters for the unbalanced agents.

Since, in this case, we are not addressing a real-world segmentation task, we do not make assumptions on the relative importance of the True Positives, True Negatives, False Positive, and False Negatives. In other words,

our task does not define which is the positive class for the problem. For this reason, we adopt the Matthew's Correlation Coefficient [197], equivalent to the Pearson's Phi Coefficient, which, differently from the Dice Score, does not rely on which of the two classes is considered the positive class.

We had one balanced agent versus 2, 4, or 8 unbalanced agents in the first test. Figures 4.2 and 4.3 shows the results for the "blob" and "checkerboard" ground truth templates, respectively. For low noise levels, methods based on the entropy calculated on larger image patches perform best, while the maximum proposal is preferable to the method based on pixel-wise entropy. As the noise level grows, the methods based on local patches are preferable over all the other methods since they also consider adjacent pixels in determining the aggregated outcome for every pixel, increasing robustness to noise. Conversely, pixel-wise entropy methods are limited, performing similarly to the maximum proposal method. When changing the ground truth pattern from *blob* to *checkerboard*, the performances generally decrease, except for the Full-Image Entropy-based method, which seems the most robust to the shape of the ground truth.

When the agents are unbalanced on C_2, the results are similar to the previous case, as shown in figures 4.4 and 4.5. Compared with the previous case, the results are almost identical for the checkerboard template. The only exception is the case of a well-performing agent ($\mu_b = 0.9$) versus eight unbalanced agents in a low noise setting ($std = 0.05$), for which the simple mean method has a noticeable loss in performances compared to the case with less unbalanced agents; this could be because the unbalanced agents on the wrong class are so many that averaging the prediction shifts the performances toward lower values. This fact is confirmed by the low performances of majority voting, meaning that the majority of the agents are voting for the wrong class. In such a case, a method based on full-image or pixel-wise entropy is preferable since it weights most the balanced agent, which is the most confident in its predictions. When comparing the results on the *blob* template with the previous set of results, we also notice that generally, the method based on 3x3 and 5x5 patches are more affected by the unbalanced agents, and their performances are no more as good as those of the Full-Image entropy. Still, they show a slight advantage in settings with a high number of unbalanced agents and high noise levels.

Figures 4.6 and 4.7 show the setting in which half of the agents are unbalanced on C_1 and the other half are unbalanced on C_2. In this case, the differences between the methods are smaller for lower noise levels, while they are still noticeable for higher levels of σ. The best solution for a lower noise level is to use a Majority Voting approach, which shows excellent

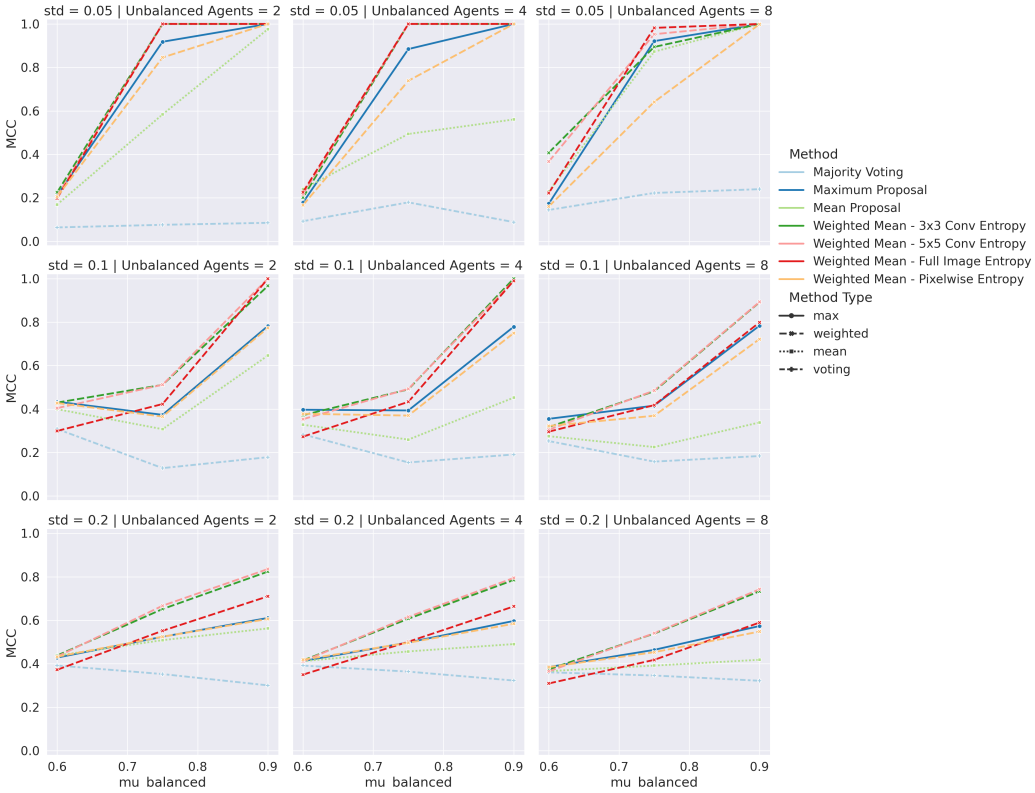


Figure 4.2: *MCC for one balanced agent versus 2, 4, 8 unbalanced agents on the class C_1 for the "blob" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output. Results for $\sigma = 0.01$ have been omitted due to True Negatives and False Positives not being part of the conflict area produced by the combination of such agents.*

performances in every case. However, as the predictions become noisy, the performances of Majority Voting drops rapidly. In such cases, methods based on 3x3 and 5x5 patch-wise entropy provide a better solution against prediction noise again, surpassing the Full-Image entropy method. Finally, as expected, the Maximum Proposal approach works best when the agents perform well on the right class while performing worse than the other methods when the agents are less expert.

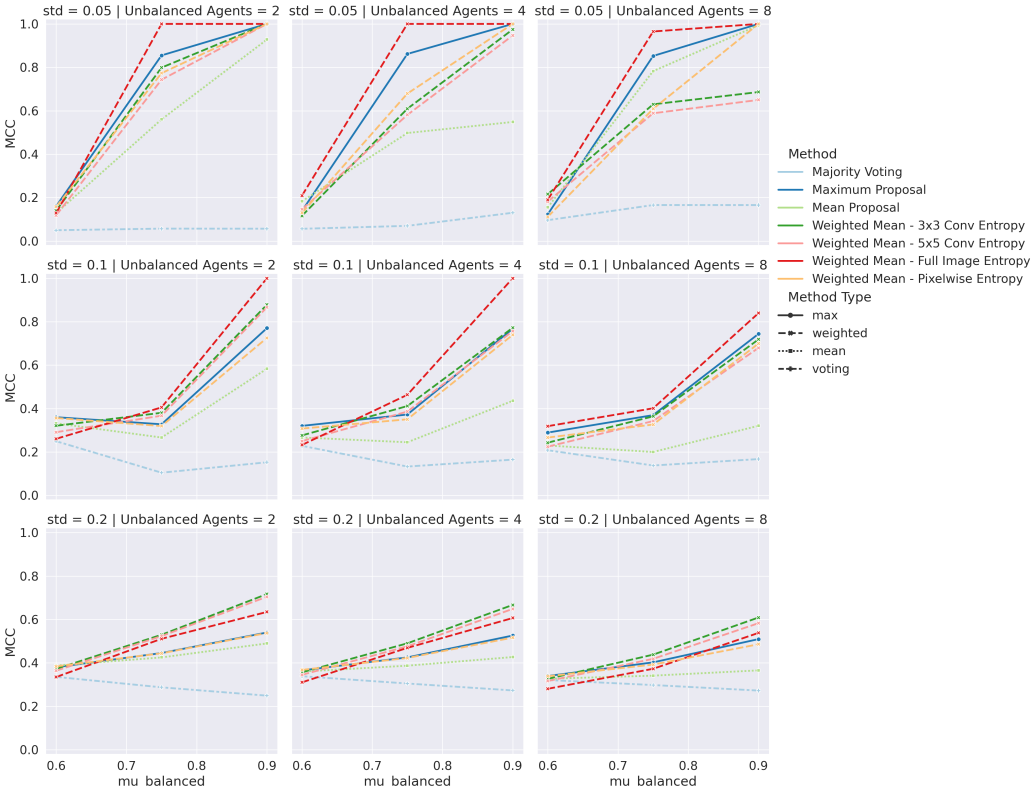


Figure 4.3: *MCC for one balanced agent versus 2, 4, 8 unbalanced agents on the class C_1 for the "checkerboard" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output. Results for $\sigma = 0.01$ have been omitted due to True Negatives and False Positives not being part of the conflict area produced by the combination of such agents.*

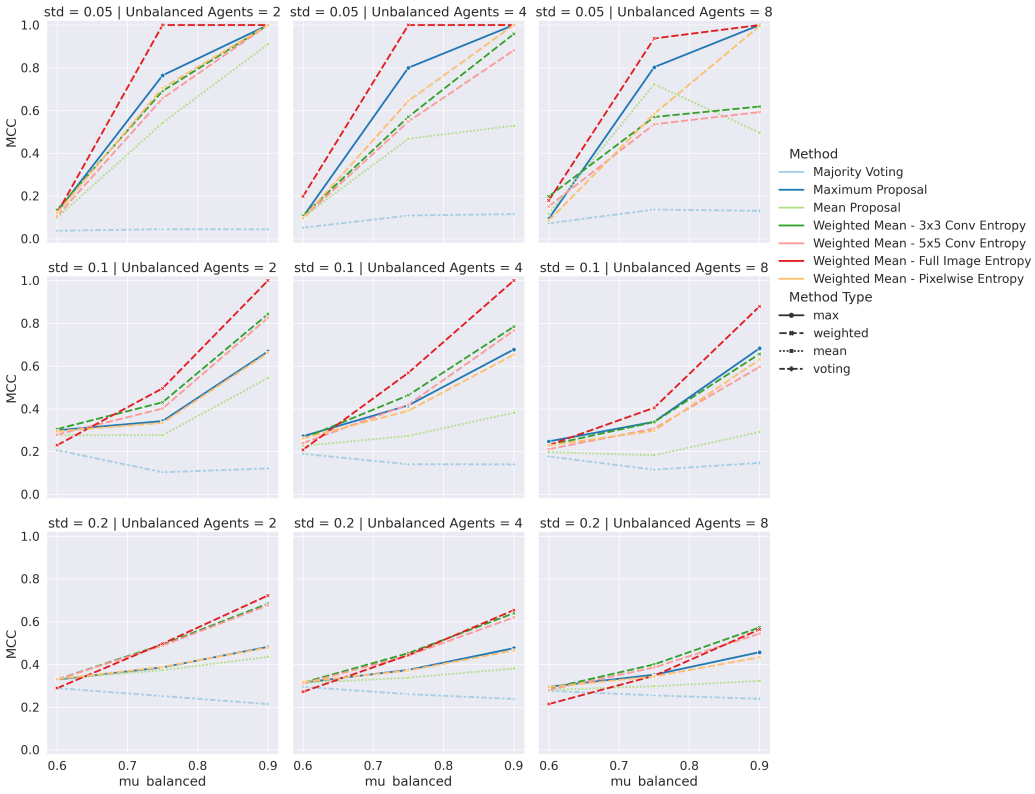


Figure 4.4: *MCC for one balanced agent versus 2, 4, 8 unbalanced agents on the class C_2 for the "blob" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output. Results for $\sigma = 0.01$ have been omitted due to True Negatives and False Positives not being part of the conflict area produced by the combination of such agents.*

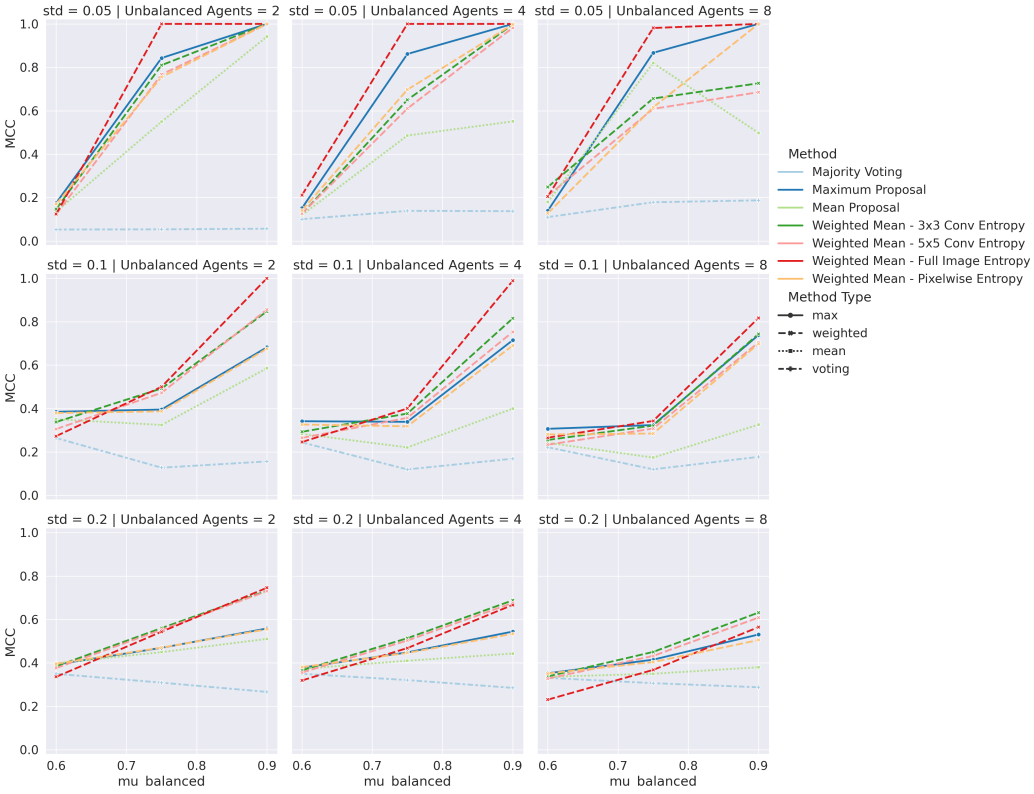


Figure 4.5: *MCC for one balanced agent versus 2, 4, 8 unbalanced agents on the class C_2 for the "checkerboard" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output. Results for $\sigma = 0.01$ have been omitted due to True Negatives and False Positives not being part of the conflict area produced by the combination of such agents.*

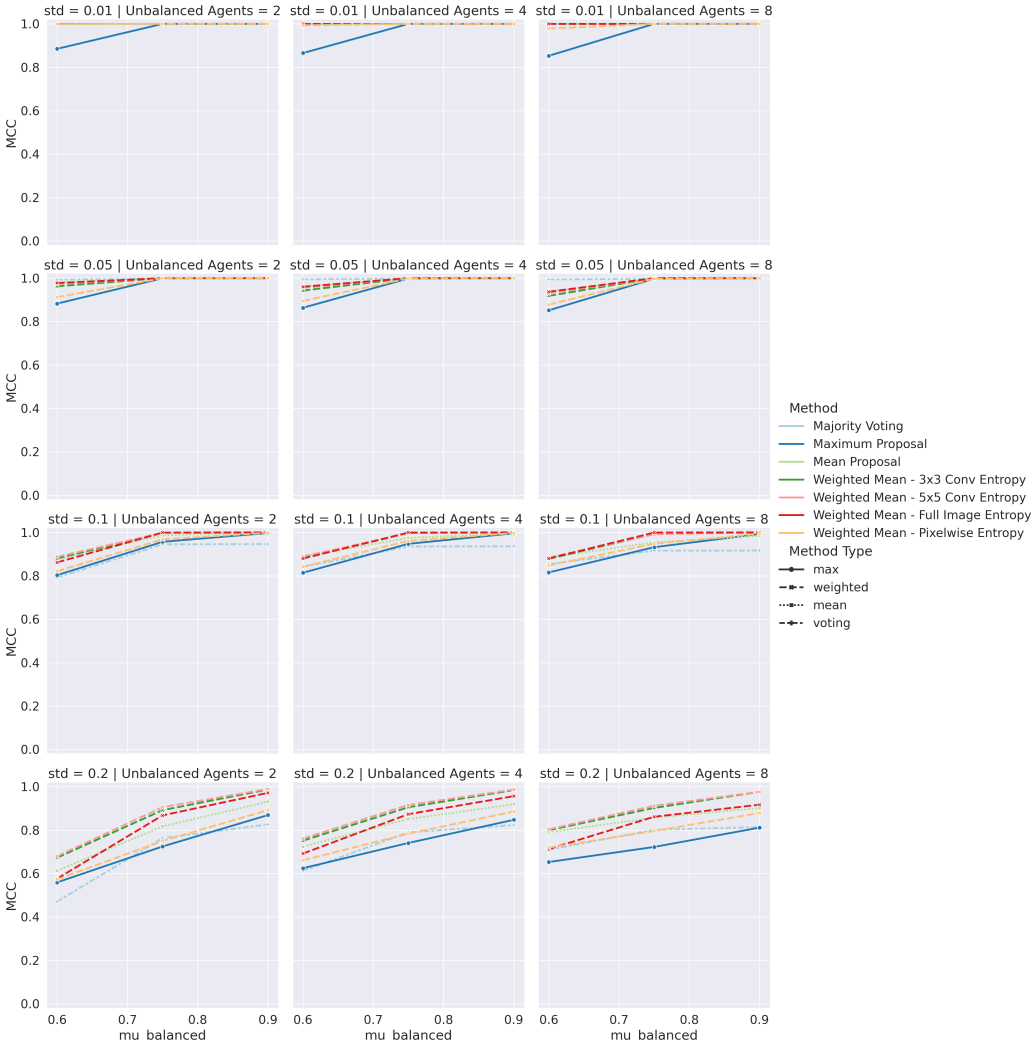


Figure 4.6: *MCC for one balanced agent versus 1, 2, 4 agents unbalanced on the class C_1 and 1, 2, 4 agents unbalanced on the class C_2 for the "blob" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output.*

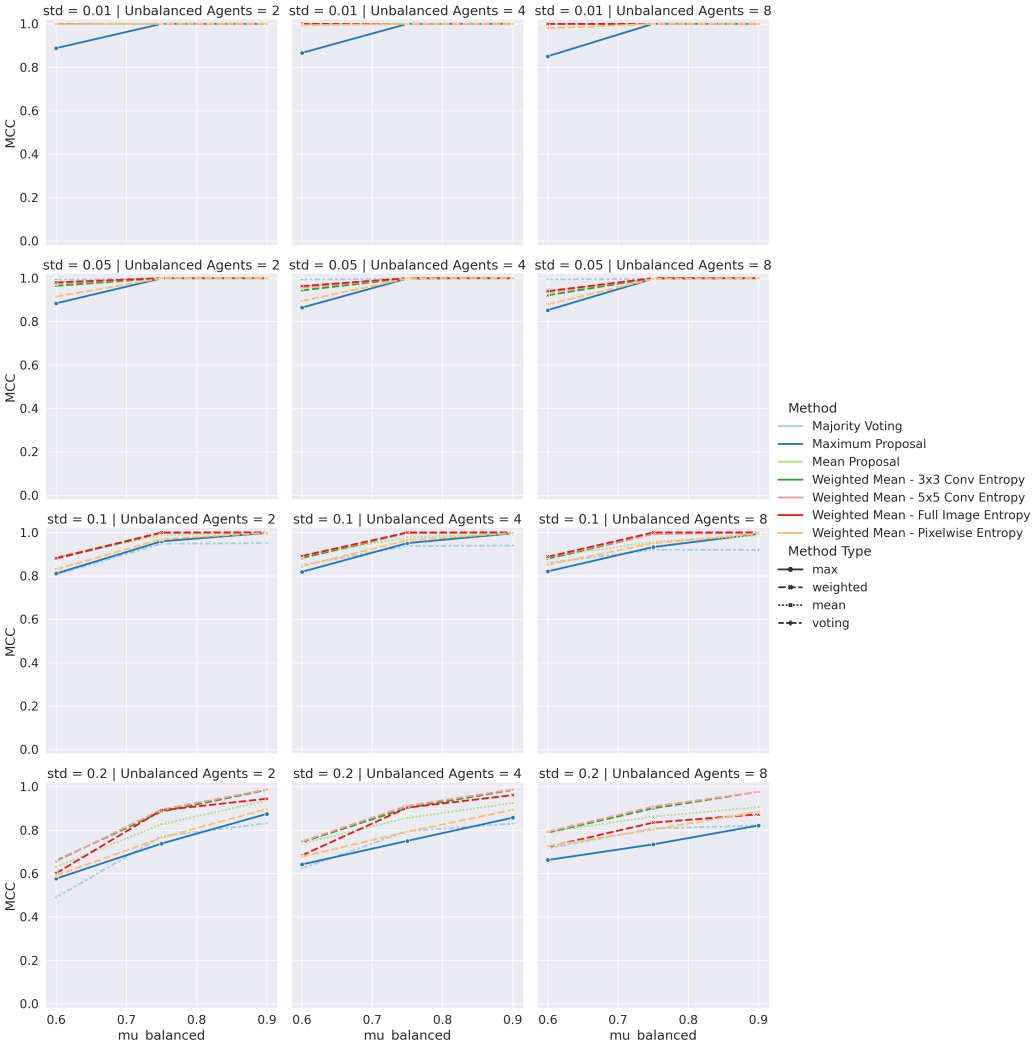


Figure 4.7: *MCC for one balanced agent versus 1, 2, 4 agents unbalanced on the class C_1 and 1, 2, 4 agents unbalanced on the class C_2 for the "checkerboard" pattern. In each graph, the horizontal axis represent the μ for the balanced agent, that is the probability to output the correct class. Each row of graphs represent a different value of σ, which introduces noise in the agent output.*

4.1.6 Experiments on Image Datasets

This section analyzes the performances of ensembling approaches in real-world scenarios. Due to the challenge of data availability in healthcare, designing deep learning models that exhibit characteristics similar to those we introduced in the previous section requires more effort. To cope with the issue of small dataset sizes, we first designed an experiment using general-purpose Images (e.g., photos of real-world objects or animals). We train four Deep Learning segmentation networks to simulate a possible real-world distributed scenario in which data is less scarce. As this experiment aims to understand how each ensembling method performs in different situations, we also included models at different stages of the training. This approach allowed us to have at disposal models with different performance degrees that could reflect scenarios in which a difficult or very specific task has to be addressed by a set of less specialized models. Finally, in the last experiment, we apply our framework to four different Deep Learning models that have been previously trained in segmenting MRI modalities from the BraTS dataset introduced in chapter 3.

Model Selection and Pre-Processing

The first experiments on real data we designed is made on the *Common Object in Context (COCO)* Dataset [161]. It is a large dataset of more than two thousand labeled images of objects belonging to about 90 categories. The size of this dataset allowed us to train Deep Learning models that are specialized in different segmentation tasks without the need for a dedicated analysis and data collection in a healthcare context. Thus, we chose four of the most represented classes of the COCO dataset, corresponding to pictures containing different animals: "(D)og", "(C)at", "(B)ird", "(H)orse", and trained four MobileNet v2 models. To introduce diversity in model performances, we trained each of the four models using different datasets, prepared as follow:

1. Starting from the original COCO Training dataset, we extracted all the images that contained *either* one of the considered labels. (e.g., we discarded the images that contained both a dog and a cat.) We obtained an intermediate dataset containing 12900 images (D: 3828, C: 3809, B: 2659, H: 2604).

2. Starting from the intermediate dataset, we split the dataset in Training, Validation, Testing (70/15/15), maintaining the class proportions constant in each dataset.

Table 4.3: *Our COCO "Unbalanced" training dataset composition*

	T_D	T_C	T_B	T_H
Dog	**1340**	446	446	446
Cat	444	**1333**	444	444
Bird	310	310	**931**	310
Horse	303	303	303	**911**

3. The *training* and *validation* datasets have been further splitted in four "expert class" datasets each (e.g. T_B,T_C,T_D,T_H, V_B, ...), in such a way that the dataset corresponding to the class "X" contains approximately 3 times the samples of class "X" with respect to the other datasets. An example for the training dataset is shown in Table 4.3

Unbalancing the datasets allowed us to train four different Deep Learning models that act as *expert agents* in the class they are trained on. We trained the models for 275 Epochs using the Categorical Cross-Entropy Loss function. To obtain a broader selection of different test cases, we considered the models trained after 1, 10, and 275 epochs for this dataset, resulting in 12 models in total with different degrees of performance across the four labels.

For the experiments on the BraTS dataset, we used the single-input SegAN-CAT architecture that will be introduced in chapter 6. For the scope of this chapter, it is sufficient to consider the architecture as an image segmentation model, which takes as input an MRI modality - i.e., T1, T1-ce, T2, or FLAIR- and produces a segmentation of the image according to one of the three labels of the dataset - ED, ET or NCR/NET-. Thus, we trained twelve different binary models for each input modality and output label combination. It is worth noting that, while sharing the same architecture and training details, these models are trained on a slightly different task from those that will be presented in chapter 6, as in this case, the models are trained to produce the ED, ET and NCR/NET, while in chapter 6 we will present models that are trained on directly classify the *whole tumor* area, already discussed in the previous chapters.

Generating the Proposals

The first step is to train and analyse the models that are obtained. Table 4.4 shows the performances of each agents on the corresponding validation sets. At this point, the ensembling step still has to be applied. Nonetheless, it could be useful to analyse the agent performances - relative to each other - to identify different learning contexts. In particular, we empirically cat-

egorised the agent performances in *Very Good* (DSC > 0.8), *Good* (0.6 < DSC < 0.8), *Average* (0.4 < DSC < 0.6), *Bad* (0.1 < DSC < 0.4) and *Failed* (DSC < 0.1). Applying this categorization we identified three different scenarios:

- Expert Agent (E1-E5): In this kind of experiment, we have one agent which performance stands out with respect to the others. In this cases we can have either a *very good* expert against either good or average models, or an *average* model against multiple *bad* or *failing* models.
- Uniform Agents (U1-U6): In this case, all the models have approximately the same performances on the task. Our experiments spans from *very good* performances -e.g., experiment U1- to *average* or *bad* models -e.g, the experiment U6-.
- Struggling Agents (S1-S3): This case can be viewed as the dual of the *Expert Agent* settings. In particular, in this setting, one agent performs poorly compared to the others.

As introduced in chapter 3, due to the differences in acquisition modalities, different labels can be easier to identify in some modalities compared to others. For example, the Enhancing-Tumor (ET) is generally more visible on the T1-ce modality, and it is reflected by the performances of the corresponding model - experiment E3 -.

4.1.7 Results

This section presents our results on the aggregation performances on the considered datasets, according to each proposed method. As discussed in the previous section, the results shown are calculated from the statistics over all the pixels belonging only to the *conflict area*. The restriction is made to avoid mixing results with the area for which an agreement already exists - in that case, we are already applying the models at the best of our knowledge, and no aggregation is needed. In other words, these results are calculated only on the parts of the image that are particularly difficult to segment for one or more models.

Table 4.5 shows the performances on the *expert agent* scenarios. In this case, the Maximum Proposal showed better performances compared to the others in three cases out of four. In experiment E1, a Majority Voting seems instead beneficial, while for E2, the Mean Weighting with Full-Image Entropy provides a significant boost compared to Majority Voting. It is worth noting that experiments E4 and E5 represent extreme cases in which only

Dataset	Output Label	Unbalanced Modality				Experiment
		Bird	Cat	Dog	Horse	
COCO ep1	cat	0.645	0.815	0.659	0.564	E2
COCO ep1	dog	0.550	0.000	0.703	0.642	S3
COCO ep1	horse	0.000	0.000	0.000	0.539	E5
COCO ep10	bird	0.645	0.000	0.000	0.000	E4
COCO ep10	cat	0.806	0.864	0.745	0.771	U2
COCO ep10	dog	0.733	0.690	0.763	0.431	S1
COCO ep10	horse	0.755	0.654	0.590	0.798	U4
COCO ep275	bird	0.761	0.688	0.699	0.503	S2
COCO ep275	cat	0.857	0.875	0.820	0.797	U1
COCO ep275	dog	0.699	0.689	0.809	0.646	E1
COCO ep275	horse	0.777	0.761	0.742	0.769	U3

Dataset	Output Label	Input Modality				Experiment
		flair	t1	t1ce	t2	
BRATS	ET	0.242	0.056	0.558	0.079	E3
	ED	0.618	0.396	0.437	0.538	U5
	NCR/NET	0.330	0.417	0.443	0.473	U6

Table 4.4: *Training of neural network models on COCO (top table) and BraTS (Bottom Table) dataset. The column* output label *indicates what the target of the particular set of models is. The columns under* Unbalanced Modality *show the Dice Score for each agent, identified by the label that has been unbalanced in its training dataset. Similarly,* Input Modality *shows the names of the BraTS models, named after the modality in input for the particular model. Lastly, the Experiment column shows the experiment identifier for the agent configuration. For COCO ep1, the row corresponding to the "bird" label is omitted as no samples with the conflicting proposal were generated.*

experiment method	E1	E2	E3	E4	E5
Majority Voting	**0.668**	0.605	0.251	0.000	0.000
Maximum Proposal	0.545	0.662	**0.533**	**0.281**	**0.693**
Mean Proposal	0.652	0.649	0.282	0.000	0.000
Weighted Mean - Pixelwise Entropy	0.650	0.659	0.283	0.055	0.550
Weighted Mean - 3x3 Conv Entropy	0.648	0.659	0.258	0.052	0.551
Weighted Mean - 5x5 Conv Entropy	0.646	0.659	0.239	0.048	0.551
Weighted Mean - Full Image Entropy	0.649	**0.670**	0.281	0.000	0.001

Table 4.5: *Experiments on the* expert agent *scenario.*

an agent can solve the task with average performances, while the other three models fail at the task. In a medical imaging scenario, this could represent the case in which an image of a rare disease or a lesion with uncommon features is submitted to a set of more general models. However, only the Maximum Proposal can solve the task in one case, although providing only a small improvement of performances over the whole image. In the other cases, even the aggregation techniques based on Entropy Weighting - except for the one computed on the whole prediction - provide acceptable performances. This suggests that the performances of each method do not depend solely on general model performances but also the confidence of each classifier and deserve further investigation.

Table 4.6 shows the performances in the uniform agent scenarios. When agents have similar performance, the Maximum Proposal is no more the best approach. Instead, the Majority Voting seems to be a better approach in half of the considered cases. In experiment U1, where all the agents have very good performances, a Mean Proposal gives a slight boost in performance upon the Majority Voting. As the mean performances of the agents decrease, our proposed methods based on Entropy Weighted Mean provides to be more effective than simpler methods, in particular averaging the entropy on smaller spatial patches constitutes a better strategy.

Lastly, we analyse the performances on the *struggling agents* scenario. In this case, 4.7 shows that the Majority Voting provides a performance boost over the other solutions is two experiments out of three. In S1, 3 agents performed in average above 0.6 Dice Score, while one performed around 0,4. In this case the three best strategies are Majority Voting, Mean Proposals and a Weighter Mean based on the entropy of the full prediction. In S2, the spread between the good-performing agents and the struggler is less severe. Being more similar to an uniform agent scenario, in

experiment method	U1	U2	U3	U4	U5	U6
Majority Voting	0.745	**0.680**	**0.673**	**0.655**	0.498	0.445
Maximum Proposal	0.654	0.594	0.569	0.562	0.369	0.405
Mean Proposal	**0.749**	0.678	0.663	0.637	0.469	0.467
Weighted Mean - Pixelwise Entropy	0.747	0.657	0.658	0.595	0.510	**0.475**
Weighted Mean - 3x3 Conv Entropy	0.746	0.656	0.656	0.591	**0.513**	0.470
Weighted Mean - 5x5 Conv Entropy	0.745	0.654	0.656	0.587	0.510	0.465
Weighted Mean - Full Image Entropy	0.746	0.668	0.650	0.627	0.489	0.458

Table 4.6: *Experiments on the* uniform agent *scenario.*

experiment method	S1	S2	S3
Majority Voting	**0.720**	**0.680**	0.672
Maximum Proposal	0.587	0.443	**0.752**
Mean Proposal	0.694	0.664	0.740
Weighted Mean - Pixelwise Entropy	0.649	0.655	0.741
Weighted Mean - 3x3 Conv Entropy	0.647	0.653	0.740
Weighted Mean - 5x5 Conv Entropy	0.646	0.653	0.740
Weighted Mean - Full Image Entropy	0.688	0.668	0.712

Table 4.7: *Experiments on the* struggling agent *scenario.*

S2 the Maximum Proposal performance decrease, similarly to what previously seen. Instead, the performances on Entropy seems to be more consistent. Finally, in S3 we have an agent that fails at the task. In this case our proposed Weighted Mean using the entropy on the full image provides a significant boost in performances.

As suggested during the discussion on the previous experiment, there may be more factors that influence the aggregation performance other than the score of the models on their validation set.

To conclude the analysis, we took a different perspective on the sets of proposals that we have generated. Instead of classifying the experiments according to the balance of agent performances in every set of aggregations, we defined two new dimensions to explore.

The first is related to the confidence of the agent. By analyzing the distribution of the confidence values for each model over their respective datasets, we noticed that as the number of epochs rises, their confidence becomes increasingly close to the unitary value; this means that the models become heavily polarized in their outputs. In other words, they tend

to output values close to either 1.0 or 0.0, regardless of the correctness of their prediction. This is unsurprising, as machine learning models are commonly trained with loss functions that encourage the network to output values close to the labels. However, it could be beneficial to understand how the ensembling methods perform according to the level of agent confidence. Again, we assume that an agent is confident in its prediction if its output is close to the negative or positive label; conversely, it would predict a value closer to the mean - i.e., the value 0.5. Based on the confidence distribution for each dataset, we set a threshold that classify as *Confident* all the agents whose prediction lies in the range $[0; 0.05] \cup [0.95; 1.0]$, while the others are classified as *Non-Confident*. In this case, confidence is not a property of the agent but rather of an agent when making a specific prediction. In other words, each agent proposal for a particular image and label is classified with a confidence label. Lastly, we counted how many agents are confident in their prediction and not for each image.

The second dimension is related to the *correctness* of the proposal. In particular, for every image, each agent would either vote for the *Right* or *Wrong* label, according to the ground truth. Thus, for each image, we identified the regions - intended as a set of pixels - where every agent has proposed the right or wrong solution, regardless of the aggregation method that will be applied. Lastly, for each pixel in the datasets of proposals, we counted how many agents proposed the right solution versus how many proposed the wrong one.

Figure 4.8 shows the Dice Scores for each aggregation method obtained by considering the areas in which we have a different balance of agents that voted for the correct label versus the number of agents that are confident in their proposal. The most evident result is that the more agents are correct in their proposal, the higher the final score of the aggregation. This, however, is quite obvious, as the more agents propose a wrong result, the more difficult it is to select the correct answer for the aggregation method. However, it can be seen how in the cases 3 agents are wrong, and one is correct, the majority of methods are heavily dependant on the agent confidence. In this case, a Maximum Proposal proved to be beneficial, as the only viable solution is to consider the answer of the most confident agent. When the balance of agent performance is even, the other methods provide satisfactory results. When no agent is confident in the solution, a Weighted Mean on Full Image entropy or a simple average is beneficial. However, the performance of the former method decreases as the number of confident agent increase. When dealing with all confident agents, a Weighted Mean using smaller patches as the 3x3 Convolution up to the single pix-

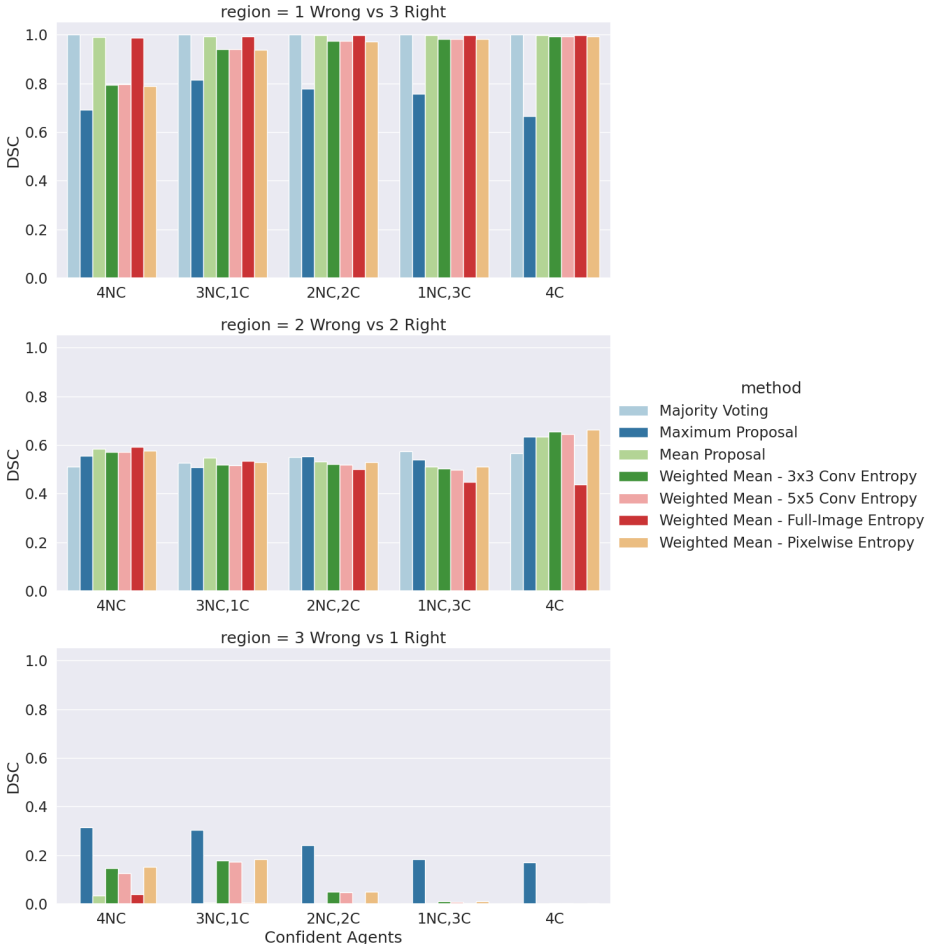


Figure 4.8: *Ground Truth Templates used for the synthetic dataset*

els seems beneficial. Lastly, if the majority of agents propose the correct solution, the Majority Voting provides the most consistent performances, followed by the Simple Mean that has the advantage of being slightly simpler to implement.

4.2 Ensembles of Heterogeneous Models

In the previous section, we show several different methods that can be used to aggregate the output of machine learning models for the segmentation task. However, some of these methods are still viable options for classification tasks. Moreover, ensembling methods are beneficial when the models are already available as they allow to use them directly without a time-consuming retrain of multiple models. To complete the analysis of the design of an ensemble learning approach, we also have to consider the first step of building a set of machine learning models on each node's private dataset. Thus, this section presents a possible approach for training different models for Automated Chest X-Ray Diagnosis and uses the already introduced ensembling techniques to solve this task.

4.2.1 Overview

Inspired by the work of Pham et al.[165], we trained seven different CNNs on the Chexpert dataset that will be used in the following chapters to study different kinds of approaches. Aside from the training of machine learning models, this particular dataset poses additional challenges that can be interesting when dealing with healthcare datasets. In particular, we first describe one possible approach to deal with label uncertainty and dependency described in Chapter 3. Then, we proceed with the description of training different Convolutional Deep Neural Networks on the CheXpert data. Finally, we show how ensembling methods could produce a single final model used in a collaborative machine learning setting.

4.2.2 Dealing with uncertain labels

As described in chapter 3, the CheXpert dataset includes a significant number of samples that have been labeled as uncertain. The uncertainty could reflect both an unreliable diagnosis or an ambiguous report used when generating the dataset. In [164], Irvin et al. investigated the issue of uncertain labels by comparing different policies, such as assuming uncertain labels as either positive or negative. Pham et al. [165], however, showed that these policies could result in wrong labels that could be misleading for the machine learning model during the training phase. For this reason, we used the *Label-Smoothing Regularization* (LSR) approach introduced by Szegedy et al. [202], which allows us to exploit a large number of uncertain labels in the CheXpert dataset while also preventing the model from becoming overconfident on uncertain samples. This is achieved by replacing the uncertain

label, u, with a random value drawn from a uniform distribution $U(a, b)$ (with $b > a > 0.5$).

4.2.3 Exploiting dependencies between labels

As described in Chapter 3, the labels included in the CheXpert dataset have a hierarchical dependency. This, it is possible to take advantage of such dependencies during the training of a classification model to achieve better performances. Inspired by the approach of Pham et al. [165], we employed a *conditional training* approach that aims at learning from data the conditional probabilities distribution of labels.

This approach relies upon the hierarchical dependency model depicted in Figure 3.2 and involves a two-steps training process:

1. We train our classifiers only with samples that have positive values (i.e., equal to 1) in labels that are not leaves in the label hierarchy (i.e., Lung Opacity and Enlarged Cardiomegaly as reported in Figure 3.2).
2. Second, we perform an additional training of the classifiers on the whole dataset to tune their prediction of labels at a higher level in the hierarchy.

The output of the resulting classifier can be viewed as the conditional probability that a label is positive when the *parent labels* in the hierarchy is assumed as positive if they exist. For this reason, when conditional training is employed, to predict the unconditional probability of unseen data, it is enough to apply the *Bayes' Rule*: the probability for each label is computed as the product of the classifier outputs for that specific label and all the parent labels in the hierarchy.

4.2.4 Chest X-Ray Classification with CNNs

We trained and compared seven different convolutional neural networks that differ in architectures, topology, and the number of parameters. The need for using differents architectures arises from the fact that each architecture, as also reported by previous works [165], has different performances on the different labels. More specifically, the networks considered have been described already in 3, and they are – along with the number of parameters –: DenseNet121 (7M), DenseNet169 (12,5M), DenseNet201 (18M) [167], InceptionResNetV2 (54M) [176], Xception (21M) [177] , VGG16 (15M) [178] , VGG19 (20M) [178]. We performed fine-tuning starting from networks already trained on the ImageNet dataset to speed-up training. In order to use the networks as classifiers for our dataset, we

removed the original dense layer. We replaced it with a Global-Average-Pooling (GAP) [203] layer, followed by a Fully Connected Layer that matches the number of labels in our dataset. As we discuss later, since there are no prevailing architectures, an approach based on the aggregation of different models can benefit the final model performance even when we are not considering a distributed setting.

Preprocessing

For the sake of simplicity, we trained our classifiers only on the frontal images included in the CheXpert dataset, as they were available for every patient. Then, we further split the dataset into a training set (roughly 90% of samples corresponding to $N = 189116$ samples) and a validation set (roughly 90% of samples corresponding to $N = 1911$ samples) for tuning the model hyper-parameters. Thus, we kept the additional set of 202 samples included in CheXpert as the test set to assess the final performances of our classifiers. We pre-processed the data by dropping additional information for each patient, such as sex and age. The uncertain labels were mapped into values sampled from a uniform distribution $U(0.55, 0.85)$, following the LSR approach introduced in the previous section. Concerning the CXR images included in the dataset, we processed them to reduce as much as possible any noise, such as text or irregular borders, that could affect learning performances. Accordingly, we first resized the images to 256 x 256 pixels and then applied a template matching algorithm to find a region of 224x224 containing a chest template. To match the data with the input shape of the networks, we converted the images from 1 to 3 channels (RGB), and we scaled their values in the range [0,1]. Finally, since the models have been pre-trained on ImageNet, we normalized all the images with that dataset mean and standard deviation.

Training Details

The networks are initialized using the weights provided by the Tensorflow 2.0 Keras module, discarding the classification layer while retaining the convolutional layers. To apply the conditional training approach described in Section4.2.3, we first trained the networks by only using the 23526 samples labeled as positive for all the findings that are not at the bottom of the label hierarchy (see Figure3.2). Then, we froze all the layers except the last fully connected one, and finally, we fine-tuned the networks on the whole training set of 189116 images. In both these training stages, we used the *binary cross-entropy* as loss function, and the learning rate was initially set

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121	**0.854**	**0.800**	0.891	0.920	0.917	**0.876**
DenseNet169	0.850	0.795	0.882	**0.936**	0.915	**0.876**
DenseNet201	0.834	0.791	0.881	0.917	0.925	0.870
InceptionResNetV2	0.816	0.784	0.897	0.925	0.919	0.869
Xception	0.841	0.770	0.880	0.909	0.924	0.865
VGG16	0.843	0.772	0.898	0.932	0.919	0.873
VGG19	0.843	0.769	**0.900**	0.927	**0.933**	0.874

Table 4.8: *Performances of the CNNs trained. The name of different CNN architecture is reported in the* `Model` *column. The performance is computed as the AUROC achieved on test set. The* `Mean` *column shows the average performance on all the five findings. We reported in bold the best performance for each finding and overall.*

to 1e-4, to be then reduced by a factor of 10 after each epoch.

CNN Results

To assess the performances of our classifiers, we computed the *Area Under the Receiving Operating Characteristic* (AUROC), introduced in Section 3.4. Table 4.8 shows the results achieved by each CNN trained for each one of the main five findings considered: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion. The results show that all the CNNs achieve a similar average AUROC over the five findings, and they also have very similar performances on every single finding: all the networks achieve outstanding performances on identifying Edema and Pleural Effusion, while they struggle to detect Cardiomegaly. On the other hand, the results show that, as expected, there is no single CNN that outperforms the others consistently for all the five findings: as an example, VGG19 achieves the best performance on Consolidation and Pleural Effusion and the worst performance on Cardiomegaly. For this reason, we applied three different ensembling strategies described in the next section to combine all the seven trained CNNs.

4.2.5 Ensembling Strategies for Classification

As confirmed by our results, in a multi-label classification task like the CheXpert one, it might be challenging to find a single classifiers that outperform the others on each target, and it might also happen that for some of the targets, no strong classifiers are available. In this setting, we might rely on ensembling strategies that combine several weak classifiers into a

stronger one. In particular, for this task we considered two of the ensembling techniques already introduced in Section 4.1.2, and *Stacking*, already introduced in Section 2.3.1.

Simple Averaging

The first approach we used simply consists in averaging the predictions made by the classifiers. If we call $y_i(\mathbf{x})$ the prediction vector provided as output by the classifier i for the input $(\mathbf{x})$, then the ensemble prediction $\tilde{y}(\mathbf{x})$ computed using N classifiers is:

$$\tilde{y}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{4.11}$$

The major drawback of this approach is that it assigns the same weight to all the classifiers, without acknowledging that some classifiers may outperform others for specific labels or might be just more confident of their predictions for a specific input $(\mathbf{x})$.

Entropy-Weighted Average

As we saw in the previous experiments, an alternative approach to simple averaging is to weigh each classifier by considering their confidence level. Accordingly, we can thus use the Entropy value of such variable as a measure of the classifier confidence:

$$H_k(p_{k,i}) = -p_{k,i} \log_2 p_{k,i} - (1 - p_{k,i}) \log_2 (1 - p_{k,i}) \tag{4.12}$$

where $p_{k,i}$ is the prediction of the classifier k for label i. As a result, $H_k(p_i)$ measures the level of uncertainty of the classifier k, while $(1 - H(p_i))$ might be seen as the confidence of classifier k about its prediction on label i. In an ensemble of N classifiers that provide a prediction for L labels, for each input we will get a *prediction matrix* $P = (p_{k,i}) \in \mathbb{R}^{NxL}$. Applying Equation 4.12 we can combine for each label $i$the classifiers predictions as follows:

$$y_i = \sum_{k=1}^{N} (1 - H_k(p_{k,i}))\, p_{k,i} \tag{4.13}$$

Stacking

The last aggregation approach we investigated involves Stacking [109]. This approach combines multiple classification models using a meta-classifier. Specifically, a train set is first used to train the base classifiers, then the predictions of the base models are used as features to train the meta-learner. The pseudo-code for the stacking algorithm is shown in Algorithm 1.

Algorithm 1 Stacking

procedure STACKEDGENERALIZATION
  $D \leftarrow \{(x_1, y_1), ..., (x_m, y_m)\}$ ▷ Training Dataset
  $\Phi \leftarrow \Phi_1, \Phi_2, ..., \Phi_T$ ▷ Base Classifier
  for $t \leftarrow 1, T$ **do**
    $h_y = \Phi_t(D)$
  end for
  $D' \leftarrow \emptyset$
  for $i = 1, 2, ..., m$ **do**
    for $t = 1, 2, ..., T$ **do**
      $z_{it} = h_t(x_i)$
    end for
    $D' \leftarrow D' \cup \{((z_{i1}, z_{i2}, ..., z_{iT}), y_i)\}$
  end for
  $h' \leftarrow \Phi(D')$ ▷ Learn meta-classifier
  return h'
end procedure

In principle, the meta-classifier for implementing stacked generalization can be trained using any machine learning method and should not be trained using the same data used to train the classifiers that have to be combined to avoid possible bias. In this work, we used a Random Forest classifier to apply *stacking* and trained it using the samples in the validation set. The Random Forest parameters have been empirically set as follows: *Max Tree Depth* was set to 30, *Number of estimators* was set to 1400, *Maximum Tree Depth* was set to 30, *Minimum Sample Split* was set to 5, and *Minimum Samples per Leaf* was set to 1.

CNN Ensembling Results

Table 4.9 shows the results of such different ensembling strategies on each of the five findings, along with the performance of the best CNNs for that specific finding. The results show that, except for the stacking approach, the ensembling strategies based on averaging allow achieving overall better performances, exploiting the differences among the CNNs. In particular,

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Best CNN	0.854	0.800	0.900	**0.936**	0.933	0.885
Simple Average	0.854	**0.811**	0.908	**0.936**	0.933	0.888
Entropy Weighted Avg.	**0.856**	**0.811**	**0.912**	**0.936**	0.930	**0.889**
Stacking	0.842	0.797	0.871	0.921	**0.937**	0.873

Table 4.9: *Comparison of the performances of the best CNN classifier for each finding (reported as* `Best CNN`*) and the three ensembling strategies considered. The performance is computed as the AUROC achieved on test set. We reported in bold the best performance for each finding and overall.*

the strategy based on entropy-weighted average resulted in being the best, achieving a slightly better performance overall and for all findings except for Pleural Effusion. Interestingly, Pleural Effusion is the only target that benefits from a stacking approach, perhaps suggesting that it requires a more sophisticated ensembling strategy.

4.3 Summary

This chapter proposed an overview of the ensembling methods for the Segmentation and Classification problem, studying both synthetic and real datasets and addressing different tasks. This technique is advantageous when machine learning models have already been trained and must be used in a collaborative setting. The advantages also apply when the computation is hosted by a single entity that holds multiple different models at disposal. In a distributed setting, this technique has the advantages of being relatively simple to implement, but they require an exchange of data, as seen in chapter 5. To investigate the potentiality of ensemble learning in a collaborative setting, we considered two different scenarios with different assumptions: In the first scenario, the heterogeneity is in the data used to train the models. In the second scenario, the heterogeneity is in the model's architecture.

In the first scenario, we proposed a set of different ensembling strategies that we considered and applied to a set of *agents*, each one corresponding to a different segmentation model. First, we proposed three more straightforward methods based on Majority Voting, Maximum Proposal, and Mean Proposal between the predictions on each image pixel. Then, we introduced four methods based on the entropy of the agent predictions, used for estimating the confidence on the agent. The four methods differ in how the entropy is spatially computed, ranging from pixel-wise up to the average confidence calculated on the whole image and applied as a weight when

averaging the agents' predictions on every pixel.

To better understand which variables in the combination of agents could affect the choice of the method, we first modeled different scenarios using a dataset of synthetic predictions, in which a single expert agent is opposed to multiple configurations of agents *unbalanced* on the two classes. Our analysis considered the number of opposing agents, the performances of the expert agent, and the noise that affects the final outcome. Our results show that our proposed methods based on entropy are effective in many scenarios.

Then, we applied the same methods on datasets of segmentation produced by real machine learning models. An interesting application scenario of the proposed ensembling strategies could be the segmentation of particular data modes, such as rare diseases. In this setting, we can suppose that only a small subset of the models are more expert than the others. For simulating such a setting, we needed to train machine learning models that followed such behavior. We first used the COCO dataset of general-purpose images to take advantage of the abundance of data, as the task is similar from a machine learning point of view. In particular, we trained different machine learning on unbalanced – i.e., heterogeneous – subsets of data. Finally, we applied our segmentation methods to the Brain Tumor Segmentation task, using four different segmentation models that take as input each one of the different modalities. In our experiments, we exploited all the combinations of labels and datasets to obtain fourteen different case studies that have been grouped in three cases: (i) Expert Agents, (ii) Uniform Agents, and (iii) Struggling agents, according to the relative model performances.

Results show that the performances of different aggregation methods may depend on a combination of variables, such as the number of agents, the relative performances of such agents, the confidence in the predictions, and the spatial shape of the segmentation. In the case of Expert Agents, the Full Image Entropy method or the Majority Voting gave good results on two different settings in which non-expert agents' performance was still acceptable. In the most extreme cases in which only an agent can solve the task, a Maximum Proposal is a successful strategy. For the uniform agent scenario, instead, Majority Voting is a good choice, surpassed by Mean Proposal when all the agents have excellent performances. When the models' performances decrease, however, the methods based on entropy averaged on smaller patches surpass the performances of simpler methods. Lastly, Majority Voting can give good performances on two cases out of three for the struggling agent scenario. As the models' performances tend

to get closer to each other, both our methods based on entropy and the Maximum Proposal give a significant performance boost.

The differences in performances between the synthetic dataset and the datasets based on real images indicate that modeling a real-world setup of machine learning agents is difficult. On the other hand, even if our analysis on images accounted for 14 different sets of machine learning models, the produced scenarios may represent particular cases not covered in our synthetic analysis. For this reason, we can consider the two analyses as complementary when choosing an aggregation method for a new system. In the last analysis, we proposed a method to investigate the impact of the *confidence* of each agent in their predictions. When using our CNN models as agents, it can be seen how the results are affected by the number of *confident* vs. *non-confident* agents, also concerning the correctness of their prediction. For example, when only half of the agents are correct in their prediction, our proposed method based on entropy works best when either none or all agents are confident. In conclusion, no aggregation method is a general solution to the problem, and the choice should be made based on the several factors cited above. Our experiment on this setting, although introductory, highlights the need for a more comprehensive study of this setting.

In the second ensemble learning scenario, we considered the problem of building a single model by accounting models with different CNN architecture. To this extent, we trained seven different Convolutional Neural Networks on the Automated Chest X-Ray Diagnosis task, using the CheXpert dataset. Since the dataset has both uncertain and hierarchical labels, we used *Label Smoothing Regularization* and *Conditional Training* to address these challenges. Our results on the trained CNNs show that no architecture can overcome the others on every label. For this reason, we adopted three ensembling techniques based on Average, Entropy Weighted Average, and Stacked Generalization to combine the classifiers into a single model. The results on the ensemble of CNNs show that our proposed Entropy Weighted Average can outperform the best CNN for each label in 4 cases out of 5. Instead, the model based on Stacking can provide better performances in the Pleural Effusion, where the other aggregation methods fail to improve the performances. The results confirm that ensembling methods can effectively combine different machine learning models into a single model, eventually showing better performances than the single parts.

In our experiments, ensemble learning proved to be particularly useful when a set of pre-trained models had to be combined and used in a collaborative setting. In particular, our proposed methods using entropy-weighting

proved beneficial in multiple scenarios. As ensemble learning is relatively simple to implement, its advantages also extend to scenarios in which a single entity has to build a solution starting from multiple models at its disposal. In a collaborative learning scenario, this technique also has the advantage of being among the simplest to implement. However, they may require the exchange of raw data, which should be managed with appropriate techniques.

However, in Chapter 6 we show how the results of this chapter can be further improved and used in combination with transfer learning. Instead, in the next chapter, we investigate the most commonly used distributed learning approach that can be used when all the nodes in the system have to start learning from scratch.

CHAPTER 5

Distributed Learning

As discussed in the first chapter, creating and maintaining large public datasets of medical information is a costly process since data sharing between different institutions needs to be approved by ethical commissions and requires the patients to be informed on how their data will be used. Adding to the cost, labeling data for machine learning requires a considerable effort from domain experts. These issues can be mitigated by adopting a collaborative -i.e., multicentric- approach. However, even when an agreement between different medical institutions is reached, other challenges must be considered. For example, health data can exhibit different modes based on the population: incidence of diseases can differ significantly by demographics and localization. Different medical institutions may be specialized in different diseases, anatomical districts, or own different imaging equipment, resulting in a shift in the distributions of individual datasets. The challenges to a collaborative machine learning approach can also be traced within the single institution that collects data since acquisition methods can differ from one medical division to another - e.g., different machinery models to acquire patient imaging.

In the previous chapters, we applied ensemble learning to different machine learning models. Ensemble learning is a valuable tool that can be

applied to models that have been already trained, even separately, by single institutions. In this chapter, however, we consider paradigms that include the advantages of distributed learning from the beginning of the training process of the models. This approach requires planning in advance and cooperation between multiple parties, while ensembling methods can be more immediate in terms of organization. As we already introduced in Chapter 2, the two main paradigms that can be applied for collaboratively training neural networks are Federated Learning and Split Learning. The two approaches are quite different and have their relative strengths and weaknesses. For this reason, a direct comparison between the two approaches could help design a potential distributed learning system. For this reason, in this chapter, we again consider the task of Automated Chest X-Ray Diagnosis to compare Federated Learning and Split Learning in a setting of data heterogeneity.

5.1 Comparison of Federated Learning and Split Learning for Healthcare Imaging

Our experiment focuses on analyzing two solutions that address the problem of collaborative learning by simulating a consortium of healthcare institutions that needs to train machine learning models for identifying a particular set of diseases. As we discussed in chapter 2, Federated Learning has already been studied in healthcare settings, and previous results show that it can achieve performances that are very close to a centralized solution [120]. Split Learning, on the other hand, has been designed for health from the very beginning, and it can provide significant advantages in terms of versatility and privacy [204, 151]. To the best of our knowledge, no other works directly compare the two methods on the Automated Chest X-Ray diagnosis task. We envision that this might be due to both the novelty of the approaches and the state of currently most-used deep learning frameworks that do not fully implement both methods natively, requiring extra effort to create a customized solution for the case at hand. Moreover, these solutions require high computational resources for the imaging field, limiting the model selection and analysis possibilities.

5.1.1 Federated Learning

Federated Learning is a new distributed learning paradigm designed by Google [112] in 2017. The key concept behind Federated Learning implementation is the decentralization of client models: the model is shared

among the various users participating in the federated system while the data remains stored within each client, ensuring privacy and confidentiality.

Let us assume a scenario in which a *federation* of N users $\{C_1, C_2, \cdots, C_n\}$ are coordinated by a central server to train a task-specific neural network. The workflow consists of 4 basic steps replicated over multiple federated rounds:

0. *Initialization:* The server trains a generic neural network to initialize the configuration. *This step is performed only in the first round*

1. *Clients selection:* A subset of clients is selected to participate in the current federated round.

2. *Models updating:* The clients download the model from the server and train it based on their local data.

3. *Reporting:* All engaged clients send the updated model to the server (i.e., weights and bias of the local model).

4. *Aggregation:* The server aggregates the received parameters and updates the global model accordingly.

5. *Repeat:* The server now owns the updated model and is ready to start again from step 1 for another round.

The global model is trained on client data without direct access to it by iterating the process for multiple federated rounds. It is useful to emphasize that weights aggregation, from a server perspective, can be done in two different ways. The first one, called *Federated Stochastic Gradient Descent (FedSGD)*, is the distributed implementation of the standard *Stochastic Gradient Descent* algorithm. It is possible to implement a FedSGD by selecting the fraction C of clients participating in a training round and a fixed learning rate η. In this way, each client *k* compute $g_k = \nabla F_k(w_t)$, the average gradients on its local data based on global model weights w at the time t. The server at this point can proceed to update the weights by considering the proportion $\frac{n_k}{n}$ of samples from clients participating in the round compared to the total samples from all clients using the formula: $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$

The second algorithm used for weights aggregation is *Federated Averaging (FedAVG)*, a generalization of the previous one. It simply consists of updating the local model several times in each client *k* before sending the weights to the server. This procedure can be helpful because, in FedSDG,

if the local clients start from the same initialization, averaging the gradients is strictly equivalent to averaging the weights themselves.

An example of Federated Learning is shown in [205]. The goal of this Federated system was to improve Google's Gboard predictive keyboard based on data and user experience; hence, the privacy and confidentiality of texts written by users via the keyboard are the focus of the project. The system's clients are the users who have the GBoard keyboard installed on their mobile phones; instead, the server is an external entity on which the global model is located. In the beginning, the server initializes a generic model to start the first federated round. Afterward, a subset of available clients for the training is selected. Then, each client device downloads the global model from the server (*Clients Selection*) and trains the model on their data as input - i.e., the text written by the user - and updates the local model (*Models Updating*). At this point, all clients send their modified model (*Reporting*) to the server, which aggregates them and creates a new representation of the global model (*Aggregation*). The new model will be sent to the clients during the next round.

5.1.2 Split Learning

The second approach we used is Split Learning, a paradigm developed in 2018 by MIT Media Lab's Camera Culture Group [204] that allows participating entities to train neural networks jointly, without sharing any raw data. Like Federated Learning, Split Learning has been developed to address model training when different institutions dealing with the same task have their collected data available. However, these are insufficient to train a neural network capable of performing well. Another interesting use case of Split Learning is when different entities or institutions hold different patient data modalities, such as electronic health records (EHR), pathological findings, and imaging data. These data taken individually are not suitable to train machine learning models; however, combining them would involve exchanging sensitive patient data, which is not always possible in healthcare due to privacy concerns. The Split Learning method for diagnosis would enable each center in this setting to contribute to constructing an aggregated model without sharing any raw data.

The key idea behind the Split Learning algorithm is that each entity (client) participating in the distributed system holds only a portion of the neural network model, from the top of the network until a specific *Cut Layer*. During the forward pass, the output from the cut layer is sent to a third entity (server) that holds the remaining network section, which is

univocal among all the clients. During the backward pass, the server can compute the gradients of its layers up to the cut layer. Then, the gradients of the cut layer are sent back to the clients, from which they can compute the remaining gradients up to the input layer. This architecture's peculiarity is that the server split of the network is trained upon all the inputs received from clients. This way, the nodes can share information with no access to raw data of the other clients.

This process can vary slightly depending on the chosen architecture - or layout -. It is possible to design a great variety of different architectures. Due to the nature of the addressed task, we focused on the *vanilla* and *U-Shaped* architectures.

Vanilla architecture

Figure 5.1a shows the most straightforward architecture of Split Learning, dubbed *vanilla* by its authors, that corresponds to the process described previously. This architecture is applicable only if the server can access the labels of the samples to calculate gradients based on the loss function. More formally, assuming to have a neural network F having N layers $\{L_0, L_1, \cdots, L_N\}$, it can be *split* in a way that there are *local* layers, i.e. localized and accessed only by the client/institution, or *shared* layers, hosted by a central server/institution. Assuming L_m is the cut layer, with $0 < m < N$, the network will be split in $\{L_0, \cdots, L_m\} \in F_{client}$ local layers and $\{L_{m+1}, \cdots, L_N\} \in F_{server}$ and shared layers. During the training phase, the output $\hat{y}_m$ of F_{client} is sent as an input to F_{server}, which completes the training obtaining the final predicted output $\hat{y}$. At this time, the server calculates the gradients ρ, obtained by the loss function $G(\hat{y}, y)$, which are sent back through F_{server} and F_{client} for the back propagation.

U-Shaped architecture

The Vanilla configuration does not ensure full privacy when the labels contain sensitive data that should not be shared with the server. The U-Shaped configuration allows clients to preserve the labels, ensuring a higher level of privacy. In this case, as shown in Figure 5.1b, there is a third part of the network F_{back}, located in the client and composed of layers $\{L_{n+1}, \cdots, L_N\}$, restricting consequently F_{server} only to layers $\{L_{m+1}, \cdots, L_n\}$. The client will take care of the computation of $\hat{y}$ and gradients ρ after receiving the intermediate output $\hat{y}_n$ from the server. This way, gradients will be sent back during backpropagation, first through the server layers and then into the client layers for weights updating. An important advantage of this ar-

chitecture is that it allows each client to define its label semantic or use a different label set from the other clients.

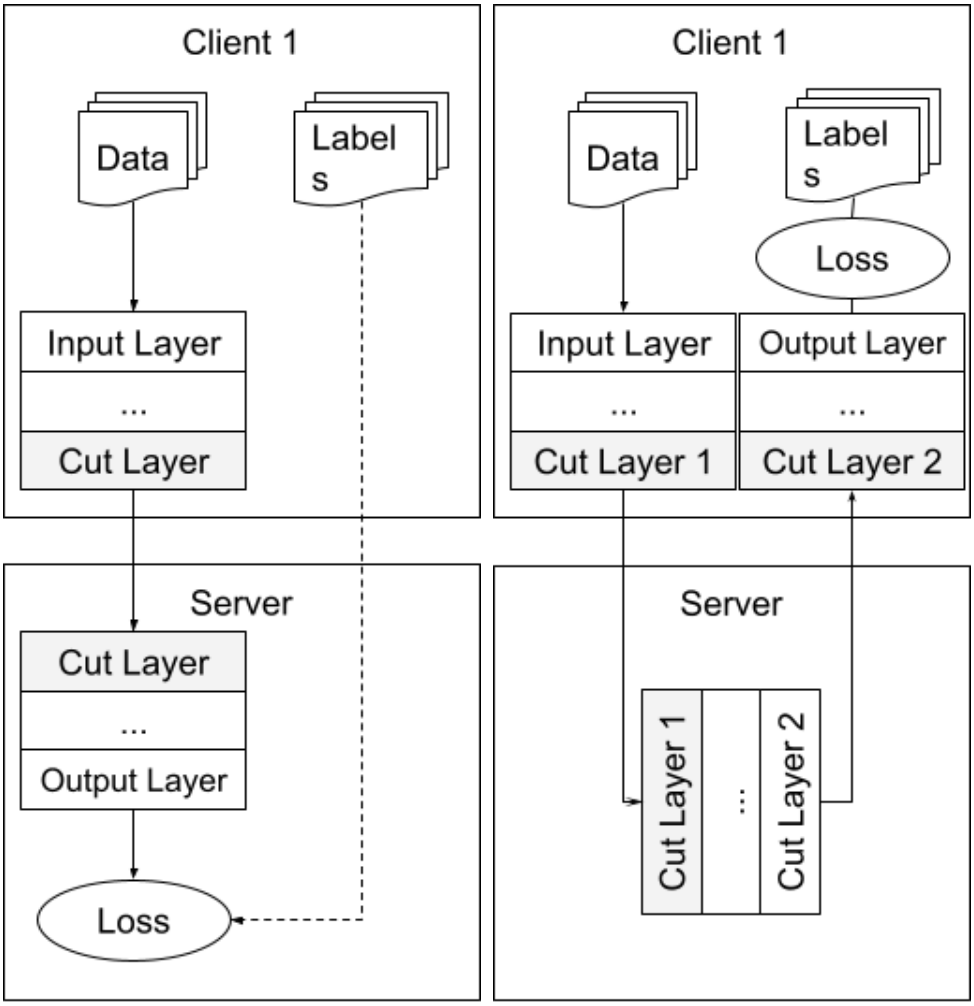


Figure 5.1: *Split Learning configurations.*

5.1.3 Experimental Design

For this experiment, we used the CheXpert dataset, already adopted in Section 4.2. We applied the same kind of pre-processing and the LSR strategy to cope with uncertain labels by assigning a randomly distributed value $x \sim U(a, b)$, with $a = 0.55$ and $b = 0.85$, to uncertain labels.

In the first experiment, we trained a centralized neural network to com-

pare the two methods broadly. This experiment will be used as a benchmark during the evaluation of the models. The training of this network was performed with the same architectural settings as the distributed scenarios. We exploited the entire CheXpert dataset using a ratio of 80% - 20% for the training/validation dataset selection. The resulting training dataset comprises 152.781 samples, while the validation dataset comprises 38.246.

Focusing on the two distributed methods experiments, we designed a distributed environment where five clients and one server participate. We have considered all fourteen pathologies during the training phase. For the evaluation, we instead focused on the five most relevant pathologies (Atelectasis, Cardiomegaly, Consolidation, Edema, Pleural Effusion) in terms of clinical relevance and amount of samples. Unlike our previous experiments on CheXpert, each client is trained on different data, while we used the same dataset for validation and testing phases. The first dimension we analyzed during the experimental campaign is the distribution of labels in the client datasets. We have partitioned the training dataset following two different ways for assigning the respective datasets to the five clients. The first is a balanced approach, in which the distribution of pathologies across them is uniform. The average number of samples for the five balanced datasets is 35645 ± 51. The second partitioning of data was done specifically to create a distribution of the five target pathologies. Each client owns a high presence of samples with a specific pathology. We ensured, whenever possible that each client holds at least 45% samples of a specific pathology. The resulting unbalanced datasets - and the corresponding clients - have been dubbed according to their majority label, resulting in the distribution: Consolidation (57.2%), Cardiomegaly (53.8%), Atelectasis (51.3%), Edema (45.8%), Pleural Effusion (34.9%). In the fifth client (Pleural Effusion), the threshold of 45% could not be reached because of the high presence of this disease in samples that were also affected by other diseases. However, the dataset of this client is still consistent with our requirements since the closest client dataset in terms of Pleural Effusion samples is the fourth, with 14.328 samples. The other clients hold circa 10.000 Pleural Effusion samples. All the clients except Edema hold an average of 35217.5 ± 292 samples, with the latter holding 37355 images. The validation dataset used in both scenarios is composed of 12.802 samples.

The second dimension to take into account during our experiments is the granularity of data sharing between clients: The *fine grained* training of the neural network on each client occurs in small, sequential, and synchronous steps: each client can only train a single batch of data before the communication with the server occurs. Then, the next client is selected to

continue the training. The first client processes the second batch of data on the next iteration, and the process goes on until the dataset is exhausted. Conversely, we define as *coarse grained* a distributed strategy in which every client performs the training on its entire dataset before the training can be performed on another client.

As the last step to define our experimental campaign, we considered the case where clients operate locally without data sharing. This test case is to assess the learning capabilities of the models when they cannot participate in a distributed system. To this extent, we trained five client models using the unbalanced datasets described above.

Implementation and Training

Due to memory limitations, we could not train a large neural network architecture such as DenseNet121 with five clients. Instead, we decided to use a MobileNet architecture [206], which is still feasible for both Federated and Split Learning, even though the final performances have been negatively affected. On the other hand, the focus of this experiment is to study the differences with different distributed learning settings rather than finding the best possible model, so we leave the trials involving a more powerful architecture as future work. Instead of training each Neural Network from scratch, we fine-tuned the MobileNet previously trained on the ImageNet dataset. At the end of the layer stack of each network, we added a Global Average Pooling followed by a new Fully Connected layer, which produces the 14-dimensional output needed for our task. Lastly, we have applied a sigmoid activation function.

Since we wanted to benchmark the model performances using each paradigm, we did not simulate a real case scenario where every client physically resides on a different machine. Instead, we implemented the distributed systems on a single instance using TensorFlow 2.5 and Google Colab. We implemented Federated Learning using the Tensorflow Federated module, which is currently in development. For implementing the coarse-grained strategy, we select all the clients during each training phase. To implement fine-grained granularity in Federated Learning, we needed to create multiple virtual clients for each actual client, each holding a single batch of data. Accordingly, we modified the client selection process and the number of rounds so that each client trained its model on one batch of data at every federated round, ensuring no overlap in training batches. Due to software limitations, we needed to simulate the server's behavior to implement Split Learning. In particular, each client owns the entire neural network, and the weights corresponding to the server split are copied on

the other clients' model whenever a server update occurs. With the fine-grained strategy, the server weights are broadcasted to the clients after each client trains a single batch of data. After that, the next client processes its batch of data. Conversely, in the coarse-grained strategy, the switch of each client is performed after the previous one has completed the training on the whole dataset.

The network has been split at the 60th layer for the Vanilla architecture, leaving 66% of the weights on the server and 33% on each client. The choice for the second cut in the U-Shaped architecture has been limited by the weights amount distribution across the network layers. To promote a fair comparison between Vanilla and U-Shaped architecture, we choose to cut the network just before the last Fully Connected Layer, which is sufficient to avoid disclosing patient labels while keeping the ratio between client and server weights almost unaltered.

For training the networks on all the experiments, we used a Stochastic Gradient Descent optimizer with a learning rate of $1e-3$. The number of epochs has been set to a maximum of 10, using Early Stopping with a *patience* factor of 4. The loss function is the binary cross-entropy loss between the ground truth labels and the outputs. The Federated Learning model training was performed using the FedSGD algorithm because it is the dual implementation of the SGD optimizer used in the training of client models in the Split Learning approach. The learning rate of clients is $1e-3$, while the server learning rate has the default value of 1. It is important to note that the learning rate of the server is used during the averaging process and therefore differs from the traditional role of a generic learning rate in a centralized system.

5.1.4 Results

In this section, we show and discuss our experimental results. First, in Table 5.1, we present our results on a Centralized model. We evaluate each proposed learning paradigm on a uniform dataset distribution and an unbalanced one. For these analyses, we compared the average model obtained with each method -i.e., each Split Learning score is the average prediction over all the resulting client models -. Then, we proceed to evaluate the performances of the client model we obtained -Table 5.2-, where applicable. Lastly, in Table 5.3, we propose an overview of the performances of the considered distributed paradigms.

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion	Mean
Centralized	0.7675	0.7135	0.8285	0.7890	0.8329	0.7863
Federated Learning, Uniform, Fine	0.7604	0.6416	0.7406	0.8549	0.8560	0.7707
Federated Learning, Uniform, Coarse	0.7165	0.7237	0.7208	0.8460	0.8384	0.7691
Split Learning, Uniform, Vanilla, Fine	0.6867	0.7086	0.7366	0.8283	0.7528	0.7426
Split Learning, Uniform, Vanilla, Coarse	0.6898	0.6991	0.7426	0.8256	0.7511	0.7416
Split Learning, Uniform, U-Shaped, Fine	0.7366	0.6507	0.6301	0.8174	0.8074	0.7285
Split Learning, Uniform, U-Shaped, Coarse	0.7094	0.6451	0.6494	0.7954	0.8057	0.7210
Local, Uniform, Client 0	0.6207	0.6432	0.7184	0.7164	0.6363	0.6670
Local, Uniform, Client 1	0.6213	0.6393	0.7033	0.7156	0.5918	0.6543
Local, Uniform, Client 2	0.5929	0.6472	0.7017	0.6984	0.6483	0.6577
Local, Uniform, Client 3	0.6275	0.6329	0.6800	0.6958	0.6150	0.6502
Local, Uniform, Client 4	0.6531	0.6379	0.6998	0.7174	0.6555	0.6727
Federated Learning, Unbalanced, Fine	0.6413	0.7147	0.7057	0.8876	0.8651	0.7629
Federated Learning, Unbalanced, Coarse	0.7910	0.6829	0.7243	0.8134	0.8145	0.7652
Split Learning, Unbalanced, Vanilla, Fine	0.7160	0.6805	0.7197	0.7868	0.7372	0.7280
Split Learning, Unbalanced, Vanilla, Coarse	0.7179	0.6902	0.6960	0.8077	0.7362	0.7296
Split Learning, Unbalanced, U-Shaped, Fine	0.7512	0.5948	0.6783	0.7580	0.8124	0.7189
Split Learning, Unbalanced, U-Shaped, Coarse	0.7504	0.5922	0.6478	0.7542	0.8040	0.7097
Local, Unbalanced, atel	0.6655	0.6287	0.6706	0.7086	0.6303	0.6607
Local, Unbalanced, card	0.5710	0.5954	0.6493	0.6124	0.4899	0.5836
Local, Unbalanced, cons	0.6511	0.6185	0.6969	0.6561	0.6166	0.6479
Local, Unbalanced, edema	0.6154	0.6205	0.6105	0.7326	0.6098	0.6378
Local, Unbalanced, peff	0.6094	0.6140	0.6732	0.7058	0.6436	0.6492

Table 5.1: *Individual label scores of the average model obtained in each experiment. The experiment's name reports the learning paradigm, the dataset distribution (uniform or unbalanced), and the data granularity (Fine or Grain). For Split learning is also indicated the architecture layout (Vanilla or U-Shaped).*

Centralized Model

To benchmark our results, we first trained a centralized model using all the available data. This centralized model achieves an average AUC of 0.7863, which is in line with our previous experiments using the same data and architecture. Analyzing the scores for the specific labels, the model has difficulty recognizing Cardiomegaly while performing well on Pleural Effusion and Consolidation, reaching an AUC of 0.83.

Uniform Datasets

The first set of experiments on distributed learning paradigms has been carried on using uniform datasets.

The first distributed paradigm we used is Federated Learning with a coarse-grained data sharing policy. In the case of uniform datasets, this approach proved to work very well and achieved performances only slightly lower than the centralized model. The average AUC is 0.7691, reaching around 0.85 for labels such as Edema and Pleural Effusion.

The Federated Learning model, trained with fine-grained sharing data, has the best classification score, reaching performances similar to a centralized model while providing the advantages that a distributed approach can offer. More specifically, the model reaches an average AUC of 0.7707 and

excellent results for the pathologies Edema and Pleural Effusion, exceeding in these cases the performance of the centralized model. However, the score on Cardiomegaly drops sharply to a value of 0.64, compared to the other two experiments discussed so far.

The first of the Split Learning experiments involves the training of a *vanilla* architecture with coarse granularity and uniform datasets. Despite a decrease in the average AUROC to 0.7416, the model is still a good classifier, showing AUCs for individual labels above 0.70 and an AUC of 0.83 for Edema. Similarly, the corresponding model with fine granularity only led to a slight difference in the average AUC (0.7426 vs. 0.7416). The AUC for individual labels stands very close to those of the previous experiment. In the following experiments, we used Split Learning with the U-Shaped architecture. When using a fine-grained sharing policy, we obtained a mean AUC of 0.7285, which is slightly lower than the *vanilla* architecture using the same sharing policy but comes with the benefit of an increase in privacy between the client and server. The individual scores show a decrease in performance for Cardiomegaly (0.6507) and Consolidation (0.6301), while performances of Atelectasis (0.7366) and Pleural Effusion (0.8074) are better than the vanilla counterpart. Even for this architecture, the AUC scores are similar to the fine-grained policy when using a coarse-grained sharing policy (0.7210 vs. 0.7285). The comparison of individual scores between the two data-sharing policies follows roughly the same pattern seen for the vanilla architecture. In conclusion, our results on the Split Learning approach suggest that Split Learning is quite robust to changes in data sharing granularity.

To highlight the importance of introducing information sharing between clients, we performed an additional experiment where we disabled client communication. In other words, every client hosts a model trained only on its dataset. The best AUC is that of *Client 4* that manages to reach a maximum of 0.6727, while the scores of other clients oscillate around 0.65. However, none of the 5 cases would represent a classifier good enough to be used in the medical practice because the risk of incorrect prediction is very high. The average of the AUCs of client models is equal to 0.6604.

Unbalanced Datasets

This section shows our results obtained by repeating the previous experiments on our unbalanced datasets. We unbalanced each client's dataset to have a prevalence of samples of a particular disease.

Even in the case of unbalanced, the coarse-grained Federated Learning model performs very well, achieving an average AUC of 0.7652 and similar

results to the centralized baseline model. Changing the dataset distribution among the clients showed a minimal effect on the performance of this distributed approach. Comparing the results with the corresponding results for the uniform dataset, we note that the AUC values for the Edema and Pleular Effusion labels are slightly decreased. However, an improvement in the Atelectasis score is also noteworthy, which increases from 0.72 to 0.79. The model trained with Federated Learning and fine granularity achieves the highest AUC, 0.887, for Edema pathology, outperforming even the centralized model. Pleural Effusion also ranks well, presenting an AUC of 0.86. These two categories are confirmed to be among the easiest to recognize within this study. The mean AUC of the model is 0.7629, which allows it to rank among the best classifiers in our experiments. On the other hand, we must report difficulties in classifying Atelectasis correctly, with an AUC of 0.64.

The results of Vanilla Split Learning with coarse granularity present a similar scenario to the balanced dataset case. We observe a slight loss of performance compared to Federated Learning. However, the resulting classifier predicts well most of the diseases, such as Edema (0.81), Atelectasis (0.72), and Pleural Effusion (0.74). Cardiomegaly and Consolidation pathologies remain the most difficult to classify, probably due to their lower representation in the dataset. Concerning the Split Learning model with fine-grained data sharing, no relevant differences are found to coarse-grained as this has an average AUC of 0.73. The best label is Edema, with an AUC of 0.79, while Cardiomegaly reaches 0.69. Recalling the same models trained with uniform datasets, we note that for both fine and coarse grain, the overall performance is slightly lower with unbalanced datasets, a sign that the change in the distribution of client datasets impacts, although only slightly, the AUC score.

When using the U-Shaped architecture, we can observe the same behavior seen in the balanced dataset scenario: the average score for both Fine and Coarse granularity are slightly lower than those of the vanilla architecture - 0.710 vs. 0.730 for Coarse Granularity and 0.719 vs. 0.728 for Fine Granularity-. Again, it is interesting to note that Pleural Effusion obtains a substantial boost in performances when choosing this architecture. At the same time, the scores on other labels such as Cardiomegaly and Edema suffer from the change of architecture. This result may indicate that some labels may be more sensitive than others to changes in the last classification layer and deserve more insights.

To complete the picture of the experiments on unbalanced datasets, we again analyzed the behavior of client models without the possibility of

Client	Paradigm	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion	Mean
Atelectasis	Local	0.6655	0.6287	0.6706	0.7086	0.6303	0.6607
Atelectasis	Split, U-Shaped, Coarse	0.6408	0.5624	0.5221	0.6333	**0.7537**	0.6225
Atelectasis	Split, U-Shaped, Fine	0.6206	0.5418	0.5408	0.6732	0.7258	0.6204
Atelectasis	Split, Vanilla, Coarse	0.6871	**0.6911**	0.6643	**0.8073**	0.7458	**0.7191**
Atelectasis	Split, Vanilla, Fine	**0.6940**	0.6612	**0.6991**	0.7638	0.7272	0.7091
Cardiomegaly	Local	0.5710	0.5954	0.6493	0.6124	0.4899	0.5836
Cardiomegaly	Split, U-Shaped, Coarse	0.6390	0.4617	0.5379	0.7475	0.7631	0.6298
Cardiomegaly	Split, U-Shaped, Fine	0.6341	0.4678	0.5425	0.7604	**0.7685**	0.6347
Cardiomegaly	Split, Vanilla, Coarse	**0.7447**	**0.6834**	**0.6805**	**0.7888**	0.7370	0.7269
Cardiomegaly	Split, Vanilla, Fine	0.7231	0.6562	0.7346	0.7874	0.7344	**0.7271**
Consolidation	Local	0.6511	0.6185	0.6969	0.6561	0.6166	0.6479
Consolidation	Split, U-Shaped, Coarse	0.6833	0.6535	0.6579	0.7275	0.6558	0.6756
Consolidation	Split, U-Shaped, Fine	0.6693	0.6631	0.6882	0.7272	0.6598	0.6815
Consolidation	Split, Vanilla, Coarse	0.6998	**0.6857**	0.7026	**0.7865**	0.7057	0.7161
Consolidation	Split, Vanilla, Fine	**0.7019**	0.6786	**0.7040**	0.7807	**0.7217**	**0.7174**
Edema	Local	0.6154	0.6205	0.6105	0.7326	0.6098	0.6378
Edema	Split, U-Shaped, Coarse	0.6331	0.5430	0.6371	0.5228	0.7509	0.6174
Edema	Split, U-Shaped, Fine	0.6417	0.5447	0.6577	0.5301	**0.7574**	0.6263
Edema	Split, Vanilla, Coarse	0.6966	0.6722	0.6649	**0.8062**	0.7146	0.7109
Edema	Split, Vanilla, Fine	**0.7124**	**0.6950**	**0.6676**	0.7802	0.7263	**0.7163**
Pleural Effusion	Local	0.6094	0.6140	0.6732	0.7058	0.6436	0.6492
Pleural Effusion	Split, U-Shaped, Coarse	0.5572	0.5264	0.5222	0.7049	0.6465	0.5914
Pleural Effusion	Split, U-Shaped, Fine	0.6168	0.5175	0.5188	0.6856	0.6658	0.6009
Pleural Effusion	Split, Vanilla, Coarse	0.6759	0.6492	0.6814	**0.7653**	0.7053	0.6954
Pleural Effusion	Split, Vanilla, Fine	**0.6951**	**0.6629**	**0.7112**	0.7528	**0.7101**	**0.7064**

Table 5.2: *Comparison of the client model for each label.*

collaboration. Table 5.1 shows that the models has generally low performances. For instance, the client trained on a prevalence of Cardiomegaly samples achieves an average AUC of 0.58, slightly higher than that of a random classifier. The average results are lower than the uniform dataset scenario, which is not always available in reality. These scores represent the results that could be achieved in a real case scenario in which each client is specialized on a particular mode of data without applying distributed training. Having set our lower bound on performances, the benefit of distributed training becomes now more evident.

As the last analysis, we compare the local models with the corresponding model trained using the different Split Learning methodologies. Table 5.2 shows that, in every case, applying Split Learning resulted in being beneficial in terms of average AUC compared to the local client performances. This result proves that Split Learning can transfer valuable information to the clients without disclosing input samples to the other participants. When considering individual label scores, the best results are obtained by applying Vanilla Split Learning with coarse or fine granularity. Unsurprisingly, U-Shaped architectures perform slightly worse than the vanilla counterpart due to the last classification layer not being shared anymore. However, it can still be proven beneficial on some target labels - such as Pleural Effusion - or when the local clients perform poorly due to their local dataset

distribution.

Paradigm Comparison

Table 5.3 summarizes the mean AUC scores for every experiment setup. As expected, the performance of the distributed approaches ranked between the upper benchmark, defined by the centralized model, and the lower benchmark, defined by the local models trained without data sharing. The Federated Learning models, achieving AUCs roughly similar to the baseline model, performed better than the Split Learning models. However, the latter provides more versatility in the model's design, and they have achieved good overall performance.

The granularity of data sharing did not affect Federated Learning nor the Split Learning paradigms to a relevant degree. Regarding the choice of datasets, the Federated Learning models do not seem to have been affected by the modification, confirming once again to be good classifiers. Instead, using unbalanced datasets in Split Learning has slightly lowered the performance of the models, which found more difficulty in the classification of diseases but still obtaining acceptable results. We envision that, in practice, this loss of performance may be compensated by designing the client shard of the network more carefully, according to the needs dictated by the individual dataset distributions. On the other hand, this may not be easy to implement in Federated Learning as the architecture is fixed for all the clients. Finally, we can point out that the easiest pathologies to classify are Edema and Pleural Effusion for almost all the models taken into account; these pathologies are also the most present in the training dataset, and for this reason, the models are more confident with their classification. Similarly, the pathology Cardiomegaly and occasionally Consolidation are the ones that in most models are more difficult to predict and may require a better choice of the architecture.

Due to the computational and data limitations in our experiment, a direct and realistic comparison of FL and SL under an efficiency perspective hasn't been possible. Since communication among nodes is an integral part of the training process in distributed learning techniques, communication efficiency plays an important role in the choice of a suitable approach. Previous works suggests that, in general, Split Learning is communication efficient when increasing the number of clients and the number of parameters. Conversely, Fedearated Learning is more efficient when increasing the dataset size, especially when the number of clients or the model size is small [207]. The authors also consider an healthcare scenario, where Split Learning performs slightly better than Federated Learning, except for

Learning Paradigm	Data Sharing Granularity	Uniform Datasets	Unbalanced Datasets
Centralized Model	-	0.7863	
Federated Learning	Coarse	0.7691	0.7652
	Fine	0.7707	0.7629
Split Learning (Vanilla)	Coarse	0.7416	0.7296
	Fine	0.7426	0.7280
Split Learning (U-Shaped)	Coarse	0.7210	0.7097
	Fine	0.7285	0.7189
Local Models	-	0.6604	0.6263

Table 5.3: *Summary of Mean AUC for each experiment.*

the case in which the dataset size is larger and the number of clients is smaller. Moreover, SL has the advantage to converge much faster than FL [110, 111].

5.2 Summary

As we discussed at the beginning of this Chapter, when pre-trained models are not available prior to design a collaborative system, a possible approach is to exploit a distributed learning technique, such as Federated Learning or Split Learning. Our results of Chapter 2 highlighted the need for a direct comparison of these two methods for a medical imaging task. To investigate this issue, we proposed a comparison between Federated and Split Learning in the context of data heterogeneity and privacy requirements.

The goals of this experiment were to: (i) compare the performances of the two methods, (ii) understand the impact of frequency of information-sharing between the clients, and (iii) investigate the impact of the data distribution in a setting of heterogeneous datasets. For this purpose, we trained several MobileNet CNNs on the CheXpert dataset. To test our approaches, we considered the five most relevant diseases in the dataset: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion.

As a reference, we first trained a CNN on the entire dataset to assess the performance of a centralized approach where privacy is not attained. Then, to simulate a distributed scenario, we split the available data into five subsets, representing different institutions holding approximately the same amount of data. In the first set of 5 splits, data is split uniformly, meaning that all clients hold the same relative amount of labels. In the second set, we artificially unbalanced each client dataset such that in each client, there is a prevalence of data corresponding to each of the target labels. To understand the impact of information sharing frequency, we considered two different granularities. In the coarse granularity, the data is exchanged after the client trained its model on the whole dataset. In the fine granularity, the clients only train a batch of data before communicating with the server. We trained a model for each client data split without using any distributed learning approach to set our baseline. Then, we trained the same MobileNet architecture with Federated Learning and Split Learning – both in Vanilla and U-Shaped layouts – using each combination of dataset balance and data granularity policy.

Our results show that both Federated Learning and Split Learning achieve performances close to centralized training. In particular, Federated Learning achieves performances roughly similar to the centralized model. While not performing as well as FL, Split Learning allowed for more versatility in model design while still achieving good performances overall. Moreover, as discussed in the previous section, Split Learning generally converge faster than Federated Learning and has slightly better efficiency in all

the considered cases, except when the dataset is very large and the number of clients is very small. This suggests that, depending on the particular setting, Split Learning could be beneficial if communication or computational resources are limited at the cost of a slight loss of performances.

As expected, both methods perform better with balanced data distribution, although they achieve good performances even with unbalanced datasets, which may prove useful in a real-world context. In all the experimental settings, distributed methods outperform models trained on the local dataset, suggesting that these methods are worthy of further investigation and might be employed in practice to deal with data privacy issues in the medical imaging field. Our results also showed a minor impact on the performances concerning how frequently clients share information with the server, limiting the amount of networking resources required when using the studied methods in practice.

In conclusion, in this chapter we showed how Distributed Learning could enable the training of Machine Learning models on multiple small datasets while ensuring privacy between the participants. This aspect is particularly relevant in healthcare, where patients' privacy is regulated by law and poses a series of practical difficulties. While this kind of approach is among the most promising, it is still the research object in the healthcare setting. Moreover, the technical complexity of its implementation may require particular technical expertise which may not be already available to smaller institutions.

CHAPTER 6

Transfer Learning

As introduced in section 1.1, the problem of data availability is a central issue in the application of machine learning models to healthcare. While medical imaging is collected daily, its availability is not comparable to other domains such as general images or text. Transfer Learning has been proved a successful approach [208] for the settings in which the amount of data is limited.

In Chapter 2 we already introduced transfer learning as a technique that allows to exploit a *source model* trained on a *source domain* for a *source task* on either a different *target domain* or a different *target task*. This usually involves either a complete or a partial retraining of the *source model* to adapt it to the *target domain/task*. Differently from what happens for language[209] and general-purpose images[210], medical field still lack of large and organized collections of pre-trained models that that can be usually applied to new tasks. Transfer learning has been successful in several fields, including image classification [211, 212, 213], natural language processing [214, 215, 216, 217], cancer subtype discovery [218], and gaming [219].

To explore the advantages of transfer learning in a collaborative learning scenario, we designed a set of experiments based on different settings.

One of the crucial aspects of Medical Imaging compared to other fields is the large variety of different tasks and analyses performed on imaging data. This variety, in turn, introduces heterogeneity in the kind of models needed for different tasks. In Section 6.1 we investigate in this direction by exploiting our results of previous chapters to show how we can train different kinds of models starting from a pre-trained Convolutional Neural Network. In particular, we exploit *embeddings*, a low-level representation of the data that can be extracted by CNNs, to expand our collection of CheXpert models with models based on trees. Moreover, we show that the resulting models can be combined with the ensembling methods introduced in Chapter 4 to improve our results further.

Then, we further validate our results by considering a real-world scenario where transfer learning is beneficial. In Section 6.2, we investigate how to exploit the knowledge acquired on CheXpert to classify samples of a smaller private dataset to perform *domain adaptation* – i.e., perform transfer learning on a dataset containing different labels than the source dataset. A transfer learning approach can be used to avoid overfitting, and at the same time, it enables generalization from one task to another [220]. However, the generalization capabilities decrease accordingly to the dissimilarity between the source and target task. In the same section, we consider another important topic of Deep Learning in medical imaging: despite having proven successful as predictive models, CNNs work as black-box models – i.e., the reasoning behind the algorithm is not interpretable by humans –. This lack of transparency raises the problem of *trust* in AI systems in such a critical setting [221]. While the experiment's aim is not focused on this problem, with the help of an explainable AI algorithm, we can give an insight into what patterns the model is focusing on for prediction.

In Chapters 2 and 3 we discussed the recent breakthrough of Generative Adversarial Model. As already discussed, adversarial models can be used to solve different tasks, including *segmentation*. An interesting application of this kind of model is to segment medical imaging. However, the different adversarial setting makes it less straightforward to perform transfer learning. To this extent, in Section 6.3, we first propose an architecture dubbed SegAN-CAT based on adversarial networks. Then, we train different models on the BraTS dataset, using either one of the four available modalities as input or all four of them. We then investigate different strategies to apply transfer learning from a source modality to a target modality.

Transfer Learning with different MRI modalities has been applied to various settings, including accelerating MRI acquisition times by applying MRI reconstruction [222] and segmentation tasks [223]. Transferring be-

tween modalities could also help deal with data heterogeneity and missing data. As we introduced in previous chapters, a relevant problem with MRI is that, in practice, they are often not available for every patient. Thus, the computer-assisted generation of the missing MRI modalities from the available ones is a problem of great interest. For example, it could enable models that require all the available modalities with limited performance loss. To this extent, in Section 6.4 we compare two different generative models based on GANs for the generation of missing modalities of brain MRIs. In particular, based on the work of Sharma et al. [64], we wanted to investigate the benefit of multi-input generative models, i.e., models that can generate a missing MRI modality from *more than one* available modality in input. To this purpose, we trained two models: MI-GAN, adapted from the approach introduced by Sharma et al., and MI-pix2pix that extends the well-known pix2pix approach introduced by Isola et al. [63]. Then, we compared the performance of these two models on the BraTS 2015 dataset. Our results show that multi-input generative models are a promising approach for the generation of missing modalities in brain MRI.

6.1 Model Training using Image Embeddings

Among the families of machine learning algorithms that are commonly used for classification, Convolutional Neural Networks are usually the first choice to process image data as the inductive bias induced by the convolution allows extracting features from images effectively. However, for many kinds of classifiers - such as those based on trees, kernels, or Bayesian statistics - it is almost impossible to learn the discriminative features directly from the image unless it is very small. It would be helpful, then, to have a compact representation of the input image that is still able to capture all the meaningful information needed to solve the task. Neural Networks can provide us with such a representation by using a technique called *Embedding*, which will be presented in the next section. It is worth noting that as the embeddings are obtained as a by-product of a pre-trained model, their information content depends on the task the model has been trained on. For example, in Auto-Encoders, a neural network model is trained to replicate the input sample as accurately as possible; hence, the embedding vectors produced by an auto-encoder aim to capture the information needed to *represent* the input sample. In our case, we extracted the embedding vectors from the networks we trained on CheXpert (Section 4), so the embeddings we obtained are tuned to represent information relevant to classify the diseases present on the CheXpert dataset.

6.1.1 Generating the embeddings

In this section, we investigate an alternative approach for classifying Chest X-Rays, by first extracting the feature vectors of a CNN classifier before the classification layer and generating a more compact representation of the input data: the *embedding vector*. When the network processes the input image, information flows through a sequence of layers to extract the meaningful features. These features are subsequently used to feed the fully connected classifier. During the feature extraction process, the image is downsampled by a pooling layer; thus, its spatial size is reduced as it flows through the layers. So, if the network is cut at a certain point, the output we observe represents the original input embedded into a lower-dimensional space. In our work, we have extracted the embeddings using all the seven neural network models we trained in Chapter 4. All the Networks have been cut at the same point, immediately after the Global Average Pooling layer, producing a one-dimensional vector. Table 6.1 shows the embedding vector shape for each model:

Model Name	Embedding Shape
DenseNet121	(1,1024)
DenseNet169	(1,1664)
DenseNet201	(1,1920)
InceptionResNetV2	(1,1536)
Xception	(1,2048)
VGG16	(1,512)
VGG19	(1,512)

Table 6.1: *Shapes of Embedding vectors generated by each CNN model.*

6.1.2 Random Forest and XGBoost on CheXpert Embeddings

Once the CNN models have generated the embeddings, we used them to train two sets of classifiers based on Random Forests and XGBoost. For each model trained in Chapter 4, we generated a dataset of image embeddings and used each dataset to train an RF and an XGBoost classifier.

For training the RF classifiers, we performed a grid search using the validation set to optimize the hyper-parameters. In particular, the hyper-parameters that we optimized are:

- *Max Depth*: the largest tree depth allowed, regulating the balance between accuracy and overfitting;
- *Min Sample Split*: the smallest number of samples to allow the split of an internal tree node;
- *Min Sample Leaf*: the smallest number of samples required for a node to become a leaf of the tree.

Due to computational constraints, we set the *Number of Estimators* - i.e., the number of trees in the forest- to 200. The *Max Features* - i.e., the number of features considered to generate a split - are set to the square root of the size of the embedding vector. The hyper-parameter optimization process was carried out for each RF classifier. Table 6.2 shows the results of the optimization and the final values of the parameters.

Concerning XGBoost, instead, we performed a hyper-parameters optimization focused on boosting the number of rounds and the maximum depth of the trees. Our analysis showed that the best settings for all the classifiers resulted in using maximum depth equal to 3 and 50 boosting rounds.

Model Name	Max Depth	Min Sample Split	Min Sample Leaf
DenseNet121	15	2	10
DenseNet169	15	2	10
DenseNet201	30	10	10
InceptionResNetV2	30	10	1
Xception	30	10	1
VGG16	5	2	10
VGG19	15	50	1

Table 6.2: *Hyper-Parameters for each Random Forest classifier.*

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
RF+DenseNet121	0.851	0.818	0.885	0.915	**0.945**	0.883
RF+DenseNet169	0.855	0.814	0.893	**0.922**	0.933	**0.884**
RF+DenseNet201	0.863	0.814	0.878	**0.922**	0.936	0.882
RF+InceptionResNetV2	0.830	0.779	0.898	0.918	0.933	0.872
RF+Xception	0.831	0.810	0.907	0.913	0.932	0.879
RF+VGG16	0.858	**0.822**	**0.913**	0.886	0.917	0.879
RF+VGG19	**0.873**	0.798	0.895	0.892	0.917	0.875

Table 6.3: *AUROC of the Random Forest classifiers on the test set. In the* `Model` *column, we reported the name of the CNN used to generate the image embeddings the RF classifier was trained from. We reported in bold the best performance for each label and we underlined the performances that are better than those of the corresponding CNN.*

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Best RF	0.855	0.814	0.893	0.922	0.933	0.884
Simple Average	0.859	**0.828**	**0.918**	**0.921**	**0.940**	0.893
Entropy Weighted Avg.	**0.872**	0.826	**0.918**	0.916	0.936	**0.897**
Stacking	0.840	0.761	0.883	0.908	0.937	0.866

Table 6.4: *Comparison of the performances of the best RF classifier for each finding (reported as `Best RF`) and the three ensembling strategies considered. The performance is computed as the AUROC achieved on test set. We reported in bold the best performance for each finding and overall.*

6.1.3 Results of Random Forests Classifiers

Table 6.3 shows the AUROC performances achieved by each RF classifiers we trained, along with the name of the CNN used for extracting the embeddings. Comparing the results to the performances of the CNN used to extract the embeddings (Table 4.8), our results show that the RF classifiers achieve in general a better performance - underlined in the table 6.3 - and this is always the case if we consider the mean performance. This result confirms, as expected, that embeddings encode all the relevant information to discriminate the findings and, more interestingly, that Random Forests are a good candidate to replace the last fully connected layer used in the network to perform classification for this task. Moreover, as for the CNNs classifiers, also in this case, there is not a single classifier that consistently outperforms all the others, suggesting that ensembling strategies can be exploited again to improve the performances. Table 6.4 compares the performances of the same ensembling strategies applied to CNN in chapter 4, this time applied to the RF classifiers. The best performance achieved by a single RF is shown for each label. The results show that the ensembling strategies consistently outperform the single best RF classifiers, and the entropy-weighted average achieved the best mean performance overall. However, simple average performs better in most of the findings. Despite the minor differences, this can be easily explained by noticing that the entropy-weighted average outperforms the simple average on identifying Atelectasis, where a single RF classifier is considerably better than all the others (see Table 6.3). in this case, the entropy-weighted average can better exploit the most confident classifier by assigning more weight to its prediction than the other classifiers. Instead, in this case, the stacking approach does not provide any improvement compared to simpler ensembling strategies.

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
XGB+DenseNet121	0.803	0.837	0.837	**0.944**	0.935	0.871
XGB+DenseNet169	0.812	0.829	0.866	0.916	0.923	0.869
XGB+DenseNet201	**0.824**	**0.867**	0.876	0.920	**0.938**	**0.885**
XGB+InceptionResNetV2	0.820	0.792	**0.911**	0.911	0.922	0.871
XGB+Xception	0.803	0.804	0.901	0.899	0.923	0.866
XGB+VGG16	0.800	0.840	0.849	0.922	0.927	0.868
XGB+VGG19	0.811	0.819	0.832	0.921	0.922	0.861

Table 6.5: *AUROC of the XGBoost classifiers on the test set. In the `Model` column we reported the name of the CNN used to generated the image embeddings the XGBoost classifier was trained from. We reported in bold the best performance for each label and we underlined the performances that are better than those of the corresponding CNN.*

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Best XGBoost	0.824	**0.867**	**0.911**	**0.944**	0.938	0.885
Simple Average	0.829	0.863	0.902	0.933	0.939	0.893
Entropy Weighted Avg.	**0.839**	0.864	0.899	0.936	**0.940**	**0.896**

Table 6.6: *Comparison of the performances of the best XGBoost classifier for each finding (reported as `Best XGB`) and the two ensembling strategies considered. The performance is computed as the AUROC achieved on test set. We reported in bold the best performance for each finding and overall.*

6.1.4 Results of XGBoost Classifiers

In the second set of experiments, we used the same approach with XGBoost classifiers. The results are shown in Table 6.5. It can be noted that, except for `XGB+DenseNet201`, the mean performances achieved with XGBoost are slightly worse or very close to the ones achieved by the corresponding CNNs. The only notable exception is the performances achieved on Cardiomegaly, where the XGBoost classifiers outperform both CNNs and the RF classifiers. This result suggests that the boosting mechanism has a more significant impact on the performances for that specific findings. Similar to what was previously done, we applied the ensembling strategies also to XGBoost classifiers. Table 6.6 shows the performances achieved with the different ensembling strategies along with those of the best XGBoost classifier – the performances of stacking strategy were worse than the one of simple and entropy-weighted averages and have been omitted. The results show that despite ensembling strategies are outperformed on some of the findings by the best single XGBoost classifier, they overall achieve a better

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121	0.854	0.800	0.891	0.920	0.917	0.876
RF+DenseNet169	0.855	0.814	0.893	0.922	0.933	0.884
XGB+DenseNet201	0.824	**0.867**	0.876	0.920	0.938	0.885
CNN Ensemble	0.856	0.811	0.912	0.936	0.930	0.889
RF Ensemble	**0.872**	0.826	**0.918**	0.916	0.936	0.897
XGB Ensemble	0.839	0.864	0.899	**0.936**	**0.940**	0.896
Final Ensemble	0.860	0.860	0.917	0.934	0.939	**0.902**

Table 6.7: *Summary of the AUROC achieved by the best classifiers and ensembles developed in this work, along with the performances of the final ensemble, computed using the entropy-weighted strategy. The performance is computed as the AUROC achieved on test set. We reported in bold the best performance for each finding and overall.*

mean performance than single XGBoost classifiers. Moreover, also in this case the entropy-weighted average is slightly better than the simple average to combine classifiers, consistently with what we previously found.

6.1.5 Final Results on CheXpert

In the previous sections, we obtained two sets of additional models for the CheXpert dataset. To obtain our final prediction on the CheXpert dataset, we combined the three ensembles of classifiers presented so far by applying the entropy-weighted average approach, which resulted in the most reliable one in this scenario.

Table 6.7 shows the performances of the final model, along with the performances of the best classifiers trained for each method – CNN, Random Forest, and XGBoost– and with the performances of the three ensembles previously presented. As expected, the results show that the final ensemble combines the benefit of the different approaches and achieves the best overall performance (AUROC value of 0.902) compared to the previously discussed approaches.

To provide one last insight into the performances of this model in practice, we present an analysis based on confusion matrices. We set a threshold for each of the five labels to discriminate among the positives and negatives when labeling unseen data. The threshold has been set as the average model output on the validation set. Moreover, to avoid misclassification of samples that are too close to the threshold, we label as *uncertain* the predictions that fall within $\pm 15\%$ of the threshold.

Table 6.8 shows the resulting confusion matrices for each finding com-

Atelectasis

Actual \ Predicted	Positive	Negative
Positive	57	10
Negative	23	87
Uncertain	25	

Cardiomegaly

Actual \ Predicted	Positive	Negative
Positive	29	31
Negative	4	127
Uncertain	11	

Consolidation

Actual \ Predicted	Positive	Negative
Positive	30	0
Negative	36	108
Uncertain	28	

Edema

Actual \ Predicted	Positive	Negative
Positive	34	3
Negative	22	124
Uncertain	19	

Pleural Effusion

Actual \ Predicted	Positive	Negative
Positive	47	13
Negative	10	123
Uncertain	9	

Table 6.8: *Confusion Matrices for the final model on each cheXpert label*

puted on the test set. This representation provides a more immediate understanding of the final performances compared to the AUROC values previously discussed. In particular, we can notice that the number of uncertain samples is approximately among the 5% and the 15% of the total, which seems to be a reasonable amount of samples requiring manual revision instead of being labeled automatically. The results also show that, in general, more false positives than false negatives are generated by the model. Although the model has not been designed with this constraint, it is desirable for a diagnostic model. The only exception is the Cardiomegaly label, suggesting a less conservative threshold might be chosen.

6.2 Transfer Learning using Embeddings on a Private Dataset

In the last section, we built a set of models for the Automatic Chest X-Ray Diagnosis task. However, our main concern is studying methods to enable learning on smaller, private datasets. In this section, we proceed further in this direction by exploiting transfer learning to train a machine learning model on a private dataset describing different labels from CheXpert, taking advantage of our previous results.

Label	Positive (%)	Negative (%)
Normal	273 (29.01)	668 (70.99)
Cardiac	93 (9.88)	848 (90.12)
Lung	427 (45.38)	514 (54.62)
Pneumothorax	38 (4.04)	903 (95.96)
Pleura	135 (14.35)	806 (85.65)
Bone	137 (14.6)	804 (85.4)
Device	147 (15.56)	794 (84.44)

Table 6.9: *Absolute frequencies of Positive, Uncertain and Negative samples for each finding (relative frequencies are reported in parentheses) in the HUM-CXR dataset.*

6.2.1 Dataset and Pre-Processing

The experiments in this section use a private dataset called HUM-CXR, collected in Humanitas Hospital in Milan, Italy. The dataset includes all the Chest X-Ray Images acquired between May, 1st 2019, and June 20th, 2019, available in the Humanitas Institutional Database. The database contained data that have been excluded in this study, specifically: 1) records not focused on the chest; 2) records without images stored in the Institutional PACS; 3) records without an available medical report; 4) records without an anteroposterior view. The Ethical Committee of Humanitas has approved this study (approval number 3/18, amendment 37/19), and due to the retrospective design a specific informed consent was granted. HUM-CXR comprises 1002 CXRs, including frontal, lateral and portable (i.e., in bed) CXRs. Each image is annotated as *normal* or *abnormal*; abnormalities were further specified as Medias, Pleura, Diaphragm, Device, Other, GI, PNX, Cardiac, Lung, Bone and Vascular, resulting in a vector of 12 labels. The labels have been manually extracted from text radiology reports. Uncertain reports were re-assessed by two independent reviewers, and discordant findings were solved by consensus. Usage of automatic labelers was unfeasible due to the Italian language in the radiology reports. Medias, Diaphgram, Other, GI, and Vascular labels were not included in this work to a limited number of x-rays available (< 30) and significant inconsistencies with CheXpert labels. Table 6.9 shows the data distribution of the labels selected for this study.

As our CheXpert models have been trained only on frontal images, we only used images from HUM-CXR tagged as *anteroposterior*, *posteroanterior* or *portable* in the DICOM descriptor, obtaining a final number of images of 941 from 746 different patients. To adapt the images to the available models, we followed the same pre-processing steps seen in the pre-

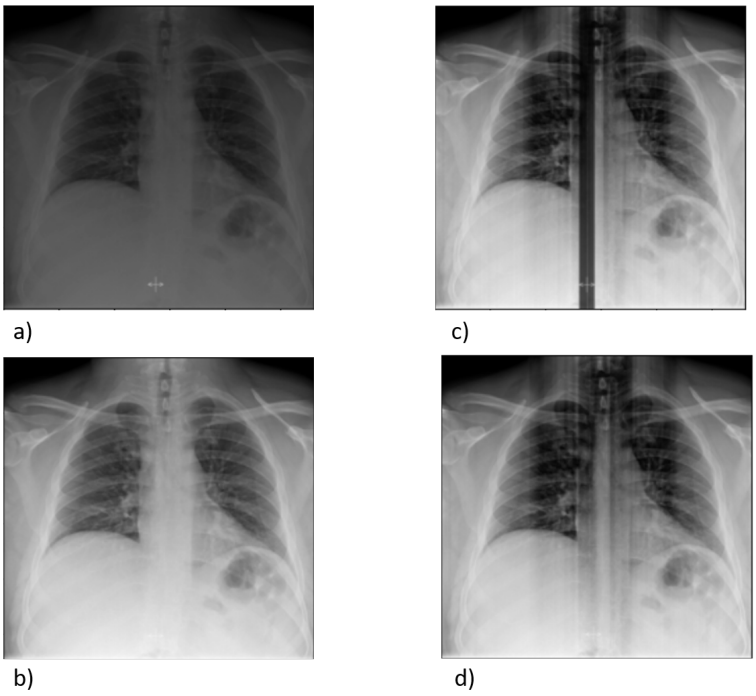


Figure 6.1: *Preprocessing with clipping the values larger than the 0.9995 quantile. The presence of a landmark, significantly more white than the other pixels, created a significant noise after normalization (a) original image (b) clipped image (c) original image normalized (d) clipped image normalized.*

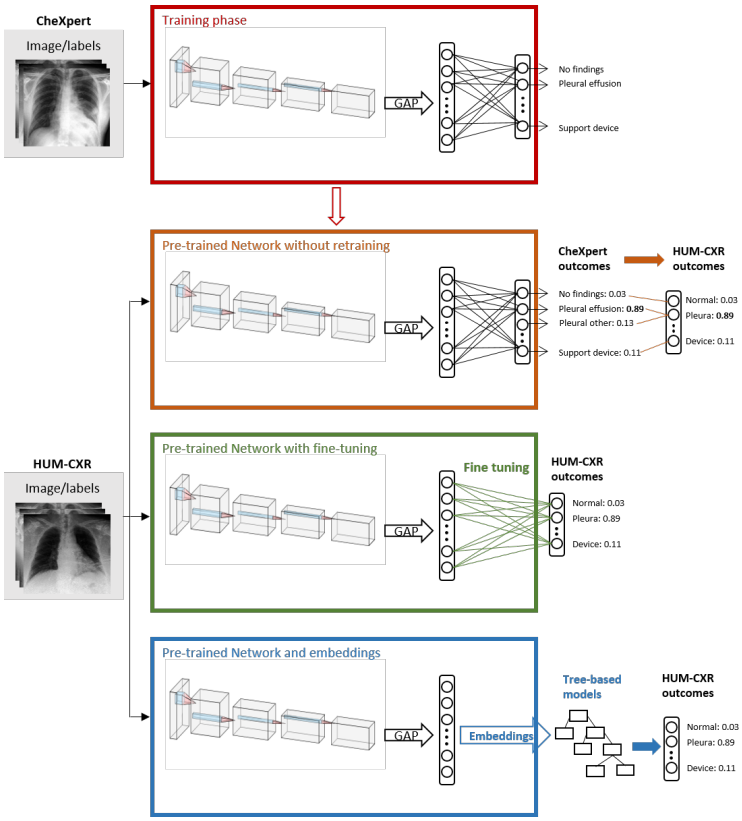


Figure 6.2: *An overview of our experimental design.*

vious study: First, we clipped pixel values with a maximum threshold of 0.9995 quantile to minimize the noise due to the landmark (see Figure 6.1). Then to match the input dimension of the models, we resized the images to 224x224 and converted the number of channels from 1 to 3 (RGB) by duplicating the input across the tree channels. Lastly, we normalized the images scaling their values in the range [0,1] and, we further standardized using ImageNet mean and standard deviation.

6.2.2 Experimental Design

In this study, we propose four different transfer learning approaches, shown in Figure 6.2. The first approach involves mapping the labels of CheXpert to the ones of HUM-CXR and just using the models pre-trained on CheXpert. The second one consists of combining the outputs of pre-trained CNNs using stacking [224]. The third one relies on exploiting the pre-

CheXpert	HUM-CXR
Pleural Effusion, Pleural Other	Pleura
Support Devices	Device
Pneumothorax	PNX
Enlarged Cardiomediastinum, Cardiomegaly	Cardiac
Lung Opacity, Lung Lesion, Consolidation, Pneumonia, Atelectasis, Edema	Lung
Fracture	Bone
No findings	Normal

Table 6.10: *Correspondence between CheXpert and HUM-CXR labels.*

trained CNNs to compute the image embeddings from HUM-CXR data and using them to train tree-based classifiers. The final one consists of tuning the CNNs pre-trained on CheXpert on HUM-CXR data. In the remainder of this section, we describe in detail these four approaches.

Pre-Trained CNNs. This approach is the most straightforward of the four and will be considered a baseline. It consists of providing the HUM-CXR images as input to the CNNs already trained on CheXpert and using the network's output to classify them based on a mapping between the labels of the two datasets. The mapping between the labels has been carefully designed after analyzing the images and the labels in the two datasets. The resulting mapping is shown in Table 6.10. When multiple CheXpert labels are assigned to a single HUM-CXR label, we selected as the final outcome for HUM-CXR the maximum model output among the CheXpert labels. As discussed in the previous section, none among the trained CNNs outperforms the others on each label. Thus, to improve the overall classification performances, we combined the outputs of the trained CNSs using the *simple average* and the *entropy-weighted average* ensembling methods. Additionally, we adopted a *stacking* approach by training a Random Forest Meta-Classifier to predict the label for HUM-CXR samples, based on the predictions of the seven CNNs trained on CheXpert, mapped to labels of HUM-CXR according to Table 6.10.

Tree-based classifiers. This approach exploits the CNNs trained on CheXpert to compute image embeddings of images included in the HUM-CXR dataset. As discussed in 6.1, image embeddings allow using much simpler models than CNNs to predict the labels of corresponding images. This approach reduces the computational resources and the data amount needed for training the classifiers, making them a suitable choice for training smaller datasets owned by single institutions. For this study, we focused on three

Model	Hyperparameters
DT	max depth=10, min samples leaf=1, min samples split=2, criterion=gini
RF	max depth=10, min samples leaf=4, min samples split=10, criterion=gini, number of estimators=100
XRT	max depth=10, min samples leaf=2, min samples split=2, criterion=entropy, number of estimators=200

Table 6.11: *Embedding models hyperparameters for the HUM-CXR dataset.*

different tree-based methods: Decision Trees (DT), Random Forest (RF), and Extremely Randomized Trees (XRT).

We first generated the image embeddings using the seven CNNs pre-trained on CheXpert. Then, for each kind of classifier, we trained seven classifiers, using 70% of the samples for training and 30% for testing. As done for the previous experiments, we applied the simple average and the entropy-weighted average to combine the seven classifiers. We tuned the training hyper-parameters of the tree-based classifiers with a grid-search optimization. The parameters are shown in Table 6.11.

Fine-tuning. Fine-tuning is the most common transfer learning approach and consists in adapting the last layers of a pre-trained neural network on a different dataset or task [220]. We removed the fully connected layer from the seven CNNs trained on CheXpert and replaced them with a seven output layer to adapt the network to the HUM-CXR labels. Then, we used the HUM-CXR dataset to fine-tune the networks for 5 epochs, using 70% of data for training, 10% for validation, and the remaining 20% as hold-out. We used early-stopping on the validation AUC, with a *patience* parameter set to 3 epochs. The loss function was a Binary cross-entropy, and the learning rate was initially set to 1e-4 and reduced by a factor of 10 after each epoch. The best performing model in the validation dataset was tested for each CNN on the hold-out set of HUM-CXR. The performances were then evaluated using simple average, entropy-weighted average, and stacking.

6.2.3 Interpretability

To address the problem of model interpretability, we applied Gradient-weighted Class Activation Map (Grad-CAM) [79], a state-of-the-art class-discriminative localization technique for CNN interpretation. This algorithm outputs a visualization of the regions of the input (heatmap) that are relevant for a specific prediction. Grad-CAMs use the gradient of an output class into the final convolutional layer to produce a saliency map, highlighting areas of the image relevant to the detection of the output class. After

producing the saliency maps, we upsampled them to the original image's dimensions and superimposed the mask on the CXR. Grad-CAM is considered an outcome explanation method, providing local explanations for each instance. Therefore, we applied Grad-CAM to randomly selected HUM-CXR. Grad-CAM heatmaps were computed for each CNN model and averaged. Besides superimposing them to the original image, we used Grad-CAM heatmaps to automatically generate a bounding box surrounding the area associated with the outcome. We created a mask with the salient part of the heatmap (pixel importance greater than the 0.8 quantiles). We used its contours to draw a bounding box highlighting the input region that mainly contributed to the prediction. DeGrave et al. [81] claimed that single local explanations are not enough to validate the correctness of the model against shortcuts and spurious correlations. Therefore, we proposed a population-level explanation averaging the saliency maps of randomly sampled 200 images where the prediction with the highest probability was selected.

6.2.4 Results

In this section, we present our experimental results. First, we show the results obtained using the pre-trained classifiers without re-training, tree-based classifiers on the extracted embeddings, and fine-tuning the classification layer. Then, we validate the performance of our models in terms of interpretability and visualization of the predicted classes. To assess the performances, we follow the same approach seen in Section 6.1 and use the AUROC as a performance score.

Pre-trained CNN without re-training. We report in Table 6.12 the performance achieved by transfer learning without re-training in terms of AUROC for each HUM-CXR class and on average. The results are shown for each CNN, ensembling with averaging, ensembling with entropy-weighted averaging, and stacking. Failures occurred mostly for Bone. Ensembling generally achieved better average results compared to single-model performance. In particular, combining the predictions with a meta-classifier (stacking) significantly improved bone classification and the mean classification AUROC.

6.2.5 Embedding

The embeddings extracted from pre-trained CNNs are used to train tree-based classifiers. Table 6.13 shows the performance achieved by DT, RF, and XRT ensembling with simple average and entropy-weighted average. The best model (RF+simple averaging) achieves a mean AUROC of 0.856

Model	Normal	Cardiac	Lung	PNX	Pleura	Bone	Device	Mean
DenseNet121	0.81	0.84	0.70	0.89	0.87	0.39	0.87	0.766
DenseNet169	0.80	0.79	0.69	0.90	0.87	0.36	0.88	0.755
DenseNet201	0.81	0.78	0.70	0.90	0.87	0.35	0.86	0.754
InceptionResNetV2	0.81	0.83	0.69	0.89	0.87	0.39	0.86	0.762
Xception	0.80	0.77	0.69	**0.91**	0.87	0.44	0.86	0.764
VGG16	0.82	**0.85**	0.70	0.89	0.89	0.41	0.86	0.775
VGG19	0.81	0.83	0.71	0.88	0.89	0.42	0.85	0.772
Averaging	0.82	0.84	0.71	**0.91**	0.89	0.38	**0.89**	0.777
Entropy	0.82	0.83	0.71	**0.91**	0.89	0.37	**0.89**	0.772
Stacking	**0.85**	0.81	**0.74**	0.88	**0.94**	**0.85**	0.84	**0.843**

Table 6.12: *CNNs results with pre-trained network without re-training in terms of AUROC. Each column represents a HUM-CXR finding. We report the results for each single network and for the three ensembling strategies. Best results for each class and in average are highlighted in bold.*

Model	Normal	Cardiac	Lung	PNX	Pleura	Bone	Device	Mean
DT+averaging	0.81	0.69	0.68	0.75	0.88	0.68	0.78	0.734
RF+averaging	**0.86**	**0.85**	0.72	**0.92**	**0.94**	**0.85**	**0.86**	**0.856**
XRT+averaging	0.85	0.84	**0.73**	**0.92**	**0.94**	**0.85**	0.85	0.853
DT+Entropy	0.81	0.69	0.68	0.75	0.88	0.69	0.78	0.753
RF+Entropy	0.85	**0.85**	0.72	**0.92**	**0.94**	**0.85**	0.85	0.853
XRT+Entropy	0.85	0.84	**0.73**	**0.92**	**0.94**	**0.85**	0.84	0.852

Table 6.13: *Results of Tree-based models trained on embeddings extracted from pre-trained CNNs in terms of AUROC. Each column represents a HUM-CXR finding. We report the results for each tree model and for both ensembling strategies. Best results for each class and in average are highlighted in bold.*

with a maximum of 0.94 for Pleura. The results show that RF and XRT achieve better classification performance than transfer learning without re-training.

6.2.6 Fine Tuning

The last set of experiments of this study consisted in fine-tuning the classification layers of the pre-trained CNNs. Single model performance improved with respect to transfer learning without re-training except for VGG16 and VGG19. Ensemble AUROC increases for all strategies. Fine-tuning combined with stacking achieves the best AUROC for PNX (0.97), while on average, it remains less performant than embeddings' best model.

Model	Normal	Cardiac	Lung	PNX	Pleura	Bone	Device	Mean
DenseNet121	**0.81**	0.73	0.76	0.94	0.90	0.73	0.78	0.807
DenseNet169	0.73	**0.88**	0.77	0.95	0.94	0.72	0.72	0.814
DenseNet201	0.83	0.81	0.71	0.94	0.94	0.74	0.82	0.828
InceptionResNetV2	**0.81**	0.86	**0.79**	0.90	0.93	0.69	0.76	0.818
Xception	**0.81**	0.82	0.73	0.94	0.95	0.68	0.80	0.819
VGG16	0.67	0.81	0.72	0.33	0.95	0.62	0.82	0.704
VGG19	0.64	0.79	0.44	0.83	0.93	0.52	**0.87**	0.717
Averaging	0.83	0.86	0.78	**0.96**	0.96	0.71	0.81	0.842
Entropy	**0.81**	0.86	0.78	0.95	0.96	0.73	0.83	0.845
Stacking	0.80	0.85	0.74	0.93	**0.97**	**0.83**	0.86	**0.853**

Table 6.14: *CNNs results with fine-tuning the classification layer of pre-trained networks in terms of AUROC. Each column represents a HUM-CXR finding. We report the results for each single network and for the three ensembling strategies. Best results for each class and in average are highlighted in bold.*

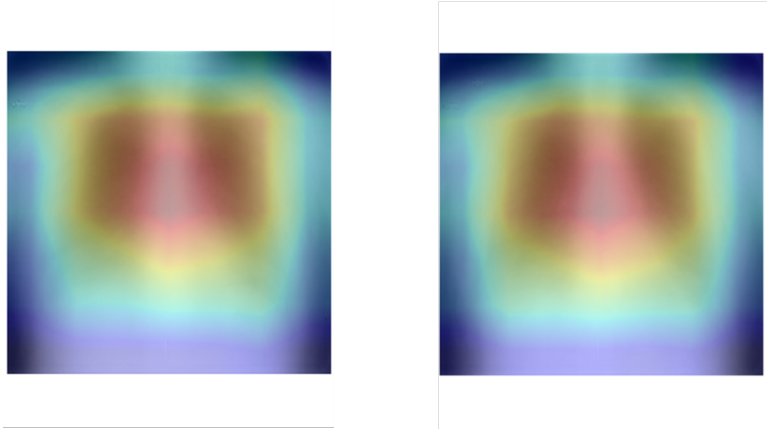


Figure 6.3: *Average of Grad-CAM saliency maps for two batches of 200 randomly sampling images.*

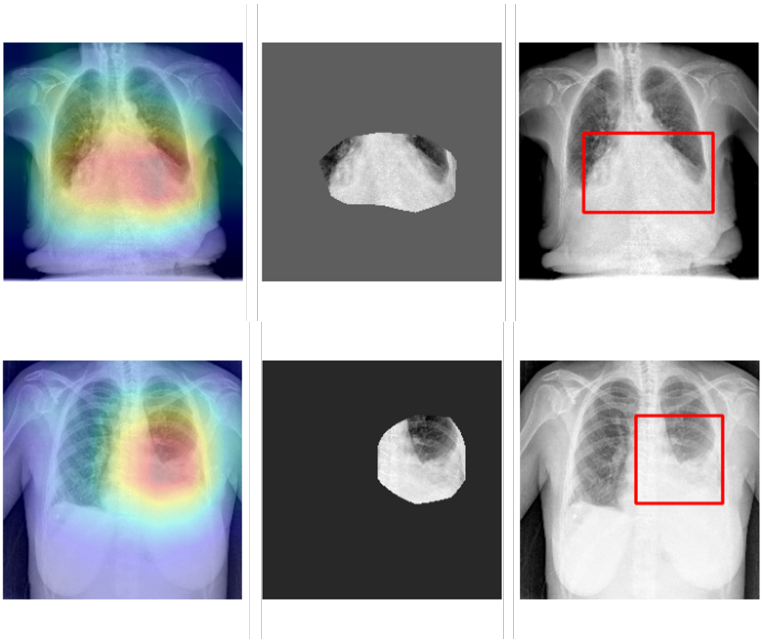


Figure 6.4: *Visualizaion of Pleura prediction maps for two selected CXRs. The three panels represent the saliency mask obtained with Grad-CAM, the relevant area (mask values larger than 0.8 quantile) and the respective bounding box.*

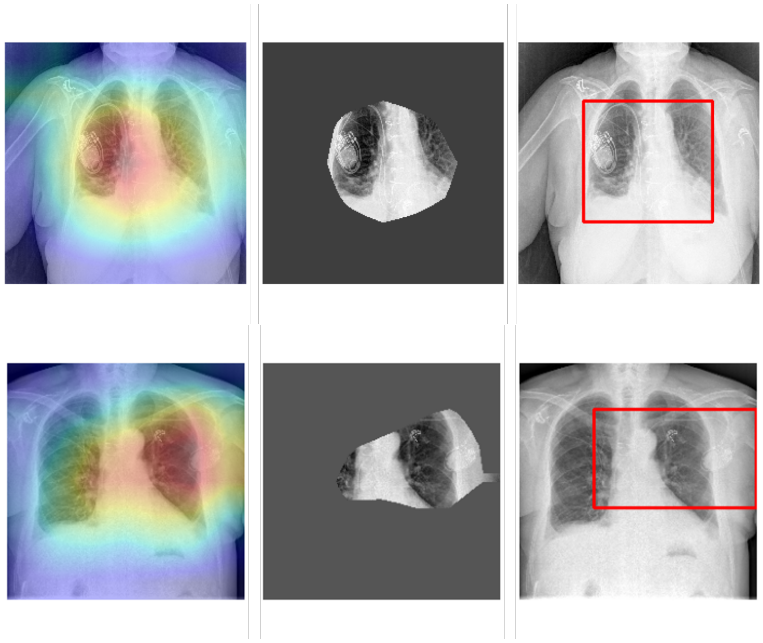


Figure 6.5: *Visualization of Device prediction maps for two selected CXRs. The three panels represent the saliency mask obtained with Grad-CAM, the relevant area (mask values larger than 0.8 quantile) and the respective bounding box.*

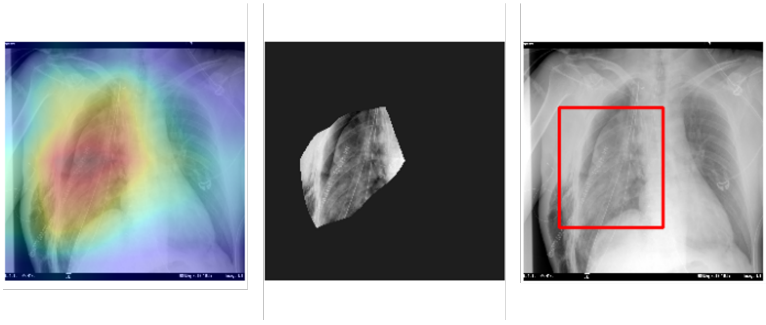


Figure 6.6: *Visualization of PNX prediction maps for a selected CXRs. The three panels represent the saliency mask obtained with Grad-CAM, the relevant area (mask values larger than 0.8 quantile) and the respective bounding box.*

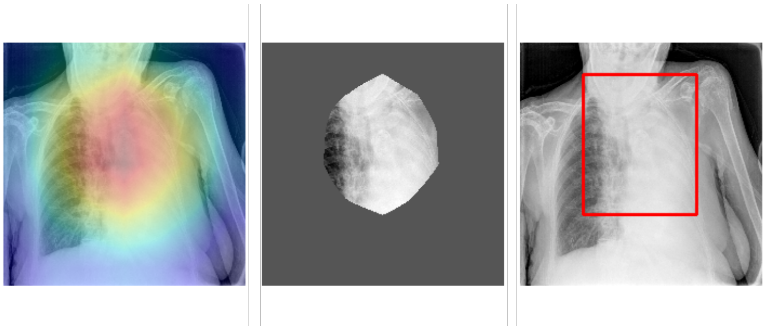


Figure 6.7: *Visualization of Lung prediction maps for a selected CXRs. The three panels represent the saliency mask obtained with Grad-CAM, the relevant area (mask values larger than 0.8 quantile) and the respective bounding box.*

6.2.7 Grad-CAM

We average the saliency maps computed with Grad-CAM of two batches of randomly sampled 200 images. Figure 6.3 proves that, at a population level, the model is generally focusing on the lung field and does not take into account shortcuts or spurious correlations that could be present in the borders. We also visualize the areas of the CXRs which the model predicts to be most indicative of each prediction and create a bounding box surrounding it. We show randomly selected examples in Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7. In Figure 6.8 we superimpose the bounding boxes for two classes to show how the model is looking at different areas of the input depending on the specific class.

XAI algorithms for visualization are successful approaches to identify potential spurious shortcuts that the network may have learned. Overall, our CNNs focused on meaningful image areas for the respective prediction. We found some inconsistencies only in some examples of device, especially with the pacemaker. Figure 6.9 shows an example of a correct classification but based on an area that does not match well the pacemaker region. The saliency map highlights the leads entering the heart as the region responsible for device prediction.

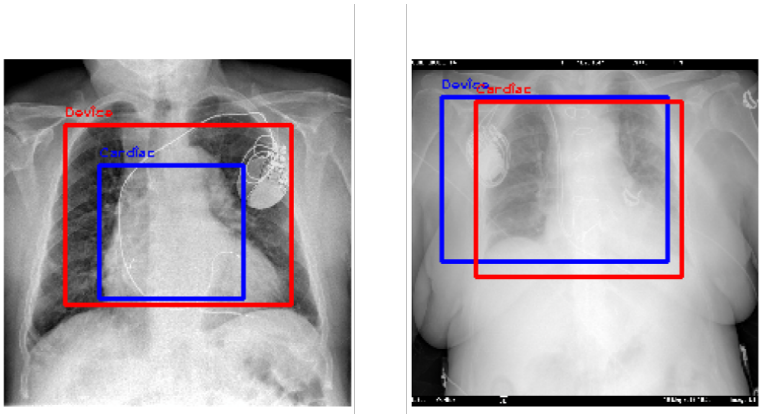


Figure 6.8: *Superimposition of bounding boxes for Cardiac and Device outcomes for two examples.*

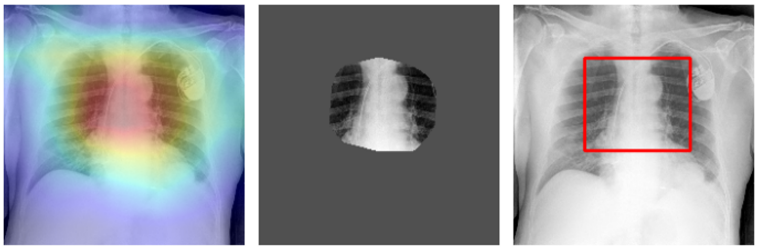


Figure 6.9: *Shortcut for the identification of a pacemaker focusing on the catheter*

6.3 Transfer Learning via Adversarial Networks

In Chapter 2 we discussed some applications of Transfer Learning applied in various settings, including medical imaging. This, as we showed in previous experiments, involves training or fine-tuning different kinds of machine learning models. However, transfer learning applied to Adversarial Networks is still an ongoing field of research. Wang et al. [225], focused on transfer learning using *WGAN-GP* [226] models and investigated how the selection of the network weights to transfer, the size of the target dataset, and the relation between source and target domain affect the final performances. Finally, in a recent work [227], Fregier and Gouray propose a new method to perform transfer learning with WGANs [228]. This section, focusing on the MRI Brain Tumor Segmentation task, investigates a different approach to perform transfer learning using an Adversarial Network. To this extent, we first introduce SegAN-CAT, an end-to-end approach to brain tumor segmentation based on Generative Adversarial Networks. Then, we study one possible application of transfer learning in a practical context. In real-world scenarios, the different Brain MRI modalities –i.e., T1, T1-ce, T2, FLAIR- might not always be all available for each patient. For this reason, we aim at training and comparing the performances of four SegAN-CAT models that use only a single modality as input. Then, we investigate whether a transfer learning approach can be used to exploit the knowledge extracted from a model, previously trained on a *source* MRI modality, to train a model that works with a different *target* MRI modality.

6.3.1 Image Segmantation with Adversarial Networks

Our approach is based on *SegAN* [185], an adversarial network architecture that was extended in two respects: (i) a *dice loss* [229] term has been added to the multi-scale loss function employed in [185]; (ii) we provide the discriminator with the input image *concatenated* to its segmentation map, while in *SegAN* the discriminator is provided with the input image *masked* using its segmentation map.

In the remainder of this section, we discuss more in detail our proposed architecture (dubbed from now on SegAN-CAT), the loss function used to train it, and the differences with respect to the original *SegAN* implementation.

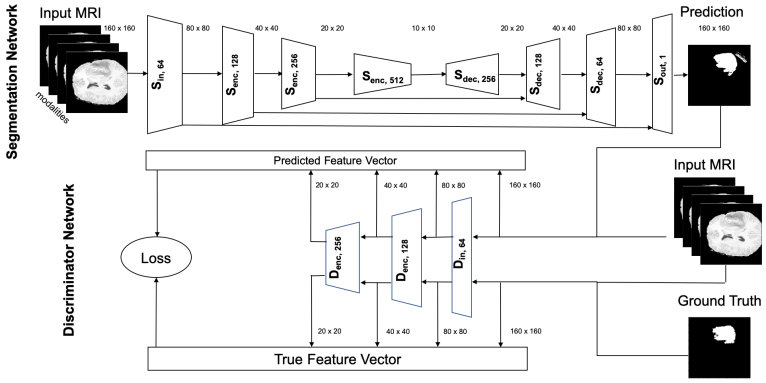


Figure 6.10: *The SegAN-CAT architecture.*

6.3.2 Network Architecture

The SegAN-CAT, shown in Figure 6.10, involves a *segmentation network* and a *discriminator network*. The segmentation network takes in input an MRI slice and produces the output the segmentation of the slice as a probability label map; the slices have size 160x160xM, where M is the number of MRI modalities used to train the model (in this experiment, M is either 1 or 4); the probability label maps have size 160x160, representing the probability for each pixel of the input slice of being part of the area of interest. Instead, the discriminator network takes in input an MRI slice and its probability label map, which can be either the one computed by the segmentation network or the ground-truth one: the slice and probability label map are either combined or concatenated together as described later in this Section; thus, the network computes a feature vector that represents a multi-scale representation of the input, which is used to compute the loss function during the training of the two networks.

Figure 6.10 shows the structure of the two networks, based on the following types of computational blocks:
(i) $S_{in,\,k}$, that is a *segmentation input block* composed of a *2D Convolution* layer with *k* filters of size 4 and
stride=2 [230, 231], followed by a *LeakyReLU* [232] activation layer;
(ii) $S_{enc,\,k}$, that is a *segmentation encoder block* has the same structure as the segmentation input block but has a *batch normalization* layer [233], before the activation layer;
(iii) $S_{dec,\,k}$, that is a *segmentation decoder block* composed of a *2D Bilinear*

Upsampling layer (factor=2) followed by a *2D Convolution* layer with *k* filters of size 3 and *stride=1*, followed by a *batch normalization* layer and a *ReLU* [86] activation layer;
(iv) $S_{out,\,k}$, that is a *segmentation output block* composed of a *2D Bilinear Upsampling* layer (factor=2) followed by a *2D Convolution* layer with *k* filters of size 3 and *stride=1*, followed by a *Sigmoid* [234] activation layer;
(v) $D_{in,\,k}$, that is a *discriminator input block* composed of a *2D Convolution* layer with *k* filters of size 4 and *stride=2*, followed by a *LeakyReLU* activation layer.
(vi) $D_{enc,\,k}$, that is a *discriminator encoder block* has the same structure as the discriminator input block but has a batch normalization layer, before the activation layer.
The convolutions weights in all *discriminator blocks* are constrained between *[-0.05;0.05]* for stabilizing the training process [228]. The output of the discriminator, indicated as *Feature Vector* in Figure 6.10, is computed by concatenating the discriminator input and the flattened output of every discriminator block. The slope for the *LeakyReLU* is 0.3; batch normalization parameters are $\epsilon = 1 * 10^{-}5$ and momentum=0.1 for both networks.

6.3.3 Discriminator Network Input

In the *SegAN* architecture, the discriminator network is given in input a *masked* slice image, computed by pixel-wise multiplication of the label probability map and each channel of the MRI slice. Instead, in SegAN-CAT architecture, the label probability map and each channel of the MRI slice is simply concatenated and provided to the discriminator network as input. While the input definition used in the original *SegAN* architecture is more compact, with the input definition we propose, it is possible for the discriminator network to extract features that also describe the area of the input MRI that is *not* included into the segmentation. As a result, we expect SegAN-CAT to have slightly better generalization capabilities than the architecture introduced in [185].

6.3.4 Loss Function

To train our networks, we use the *Multiscale Adversarial Loss*, as in SegAN [185]. This particular loss function, applied also in generative problems [235], allows to perform *feature matching* between the ground truth and the output of segmentation network, also optimizing the network weights on features extracted at multiple resolutions rather than focusing just on the pixel level. Thus, the *Multiscale Adversarial Loss* is defined as follows:

$$\ell_{\text{mae}}(f_D(x), f_D(x')) = \frac{1}{L}\sum_{i=1}^{L} ||f_D^i(x) - f_D^i(x')||_1 \tag{6.1}$$

where L is the number of layers in the discriminator network; $f_D^i(\cdot)$ is the output of the activation layer after the discriminator block at position $i+1$, e.g., $f_D^1(\cdot)$ is the discriminator input, $f_D^2(\cdot)$ is the output of the first activation layer, etc; x denotes the input of the discriminator when the label probability map is computed by the segmentation network, while x' is the input of the discriminator when the ground-truth is used.

In addition to the *Multiscale Adversarial Loss*, we introduced an additional term to the loss function defined as:

$$\ell_{\text{dice}}(g, p) = 1 - \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{6.2}$$

where g is a ground truth image, p denotes the predicted values and the sums run over the pixels of the image; the definition of ℓ_{dice} is based on a differentiable form of the *Sørensen-Dice* coefficient [229] – already introduced in Chapter 3 –, it ranges between 0 and 1, and accounts for the overlap between the segmented areas in the ground-truth and in the output of the segmentation network. The use of this term, which allows to assess the quality of the generated segmentation maps, should result in a more stable training process and eventually in a model with better performances.

Overall, the complete objective function used to train the networks is defined as:

$$\begin{aligned}\min_{\theta_S}\max_{\theta_D} \frac{1}{N}\sum_{n=1}^{N} &\ell_{\text{mae}}(f_D(x_n \circ S(x_n)), f_D(x_n \circ y_n)) \\ &+ \ell_{\text{dice}}(y_n, S(x_n))\end{aligned} \tag{6.3}$$

where x_n is an input MRI, y_n is the ground truth, and the $\circ$ operator is either a concatenation on the channel axis in SegAN-CAT or a pixel-wise multiplication in *SegAN* (as illustrated in Section 6.3.3).

6.3.5 Transfer learning with Adversarial Networks

To test our approach, we compared the SegAN-CAT and the *SegAN* architecture on the BraTS segmentation task, using both the BraTS 2015 [162] and the BraTS 2019 [31] datasets. As the images in both datasets have an isotropic resolution of 1mm^3 per voxel, we do not perform any further spatial resampling. Following the SegAN implementation [185], we

center-crop each MRI to a 180 x 180 x 128 volume in order to remove black regions while keeping all the relevant data. For each MRI volume, we clip voxel values to the 2^{nd} and 98^{nd} percentile to remove outliers. Then we apply Feature Scaling [236] to normalize the intensity range between 0 and 1. Finally, we split the data in Training/Validation sets (respectively of size 80% and 20%) using stratified sampling to keep balanced HG and LG subjects within each subset. For Brats 2019, we also apply stratified sampling based on the institution that provided the data.

To evaluate our results, we use *precision*, *sensitivity*, and *Dice Score*. To compute these metrics, we use a threshold of 0.5 on the output of the segmentation network in order to obtain a binary classification for each pixel. Although our models work on a single MRI slice at a time, we compute the metrics by considering the TP, FP, TN, and FN on every slice of the same MRI volume. Model selection is performed according to the dice score value. It allows accounting for both False Positives and False Negatives, leading to a better assessment of the segmentation quality compared to the precision and sensitivity scores.

To transfer a SegAN-CAT model trained from MRIs that include a single modality (*source*) to train a model from MRIs that include a single but *different* modality (*target*) on our SegAN-CATmodel, we used the most straightforward approach: we took the segmentation and discriminator networks trained on the source modality and *fine-tuned* them on the target modality.

In particular, we studied two different fine-tuning processes to retrain the networks on the target modality:
(i) we fine-tuned all the weights of both the segmentation and the discriminator networks;
(ii) we fine-tuned all the weights of the segmentation network and only the weights of the input layer of the discriminator network.

Although keeping the several layers of discriminator fixed during the fine-tuning might prevent it from adapting entirely to the target domain. This solution could help retain more knowledge from the source domain while adapting the segmentation network. This choice is motivated by the fact that the discriminator is the most critical part of an adversarial network to transfer to the target domain [225].

6.3.6 Results

We trained both multi-modal models – i.e., models designed to work with MRIs that include different contrast modalities – and uni-modal models –

Model	BraTS 2015			BraTS 2019		
	Dice Score	Precision	Sensitivity	Dice Score	Precision	Sensitivity
SegAN	0.705	0.759	0.694	0.766	0.745	0.834
SegAN w/ Dice Loss	0.825	0.901	0.785	0.814	0.850	0.810
SegAN-CAT	**0.859**	0.882	0.852	**0.825**	0.842	0.835

Table 6.15: *Performance of* SegAN*,* SegAN *with dice loss, and SegAN-CAT. The results reported are the average of the dice score, the precision, and the sensitivity computed on each one of the MRI volume included in test set of BraTS 2015 and BraTS 2019. We reported in bold the best performance for each dataset.*

i.e., designed to work with MRIs that include only one specific contrast modality–.

First, we trained three multi-modal models that use all the four contrast modalities, i.e., T1, T1c, T2, FLAIR, available in the datasets: (i) a model that implements the original *SegAN* architecture, (ii) a model that implements the *SegAN* architecture with the additional dice loss term, and (iii) a model that implements the SegAN-CAT architecture thoroughly, i.e., with the additional dice loss term and the input discriminator concatenation.

Each model is trained and tested both on BraTS 2015 and BraTS 2019 datasets using a batch size of 64 slices. During the training phase, we perform data augmentation by applying *random cropping* [231] using a window of size 160 x 160, following the SegAN approach [185]. During the validation phase, we apply *center cropping* [231] to match the input size of the network, discarding most of the black border of the MRI. All the models are trained using the same initialization seed, RMSprop [237] (*lr: $2*10^{-5}$*), and *Early Stopping* [238] (*patience = 300 epochs*) on Dice Score evaluation metric. An epoch corresponds approximately to 28000 slices for the Brats 2015 dataset and to 34000 for the Brats 2019 dataset.

Table 6.15 compares the performance of all the multi-modal models trained on two datasets. The results show that both the dice loss term and the discriminator input concatenation, introduced in the SegAN-CAT, led to better performances on both datasets. Results also suggest that BraTS 2019 is slightly more challenging than BraTS 2015, leading to a lower performance of all three models.

In the following experiment, we wanted to investigate how the information content of each contrast modality affects the model performance. To this purpose, we trained four uni-modal models, each one using only a single contrast modality.

Table 6.16 compares the performance of these four SegAN-CAT models

Modality	BraTS 2015			BraTS 2019		
	Dice Score	Precision	Sensitivity	Dice Score	Precision	Sensitivity
T1	0.538	0.570	0.557	0.542	0.586	0.609
T1c	0.578	0.613	0.581	0.587	0.678	0.577
T2	0.721	0.773	0.724	0.675	0.828	0.607
FLAIR	0.810	0.858	0.787	0.763	0.757	0.810
ALL	0.859	0.882	0.852	0.825	0.842	0.835

Table 6.16: *Performance of SegAN-CAT models trained from MRIs with only one contrast modality. The results are reported for each modality as the average of the dice score, the precision, and the sensitivity computed on each one of the MRI volume included in test set of BraTS 2015 and BraTS 2019. As a reference, the performance of SegAN-CAT trained with all the contrast modalities is reported in the last row of the table.*

trained using a single contrast modality. The results show that none of the models trained with a single modality can achieve the same performance achieved by SegAN-CAT which uses all the four contrast modalities together. This suggests that none of the four modalities alone contains all the relevant information to identify the tumor. However, the model trained using only the FLAIR modality can perform much better than all the other models on both datasets. These results are not surprising as FLAIR modality allows to identify more clearly the edema and the lesions in specific areas of the brain due to the suppression of the cerebrospinal fluid in the images.

Lastly, we investigated whether it is possible to transfer a model trained on a specific contrast modality to a different contrast modality. We aimed to evaluate the similarities among the models and have an insight into how easily the knowledge learned from a modality could be transferred to the others. To this extent, we applied all the uni-modal models previously trained on each of the contrast modalities to images taken with different modalities.

Figure 6.11 shows the performance of all the uni-modal models when applied to each modality alone. As expected, all the uni-modal models reached the best performance on images acquired with the same contrast modality they have been trained for. Therefore, to transfer a model successfully across the contrast modalities, it is necessary to adapt the trained networks with data from the target domain, i.e., the target contrast modality. The results also show how models could be easily transferred across modalities that are known to be similar: models trained on T1 and T1c perform poorly on T2 and FLAIR, while they perform much better when applied to the other modality similar to the one they are trained on (i.e., T1 or T1c).

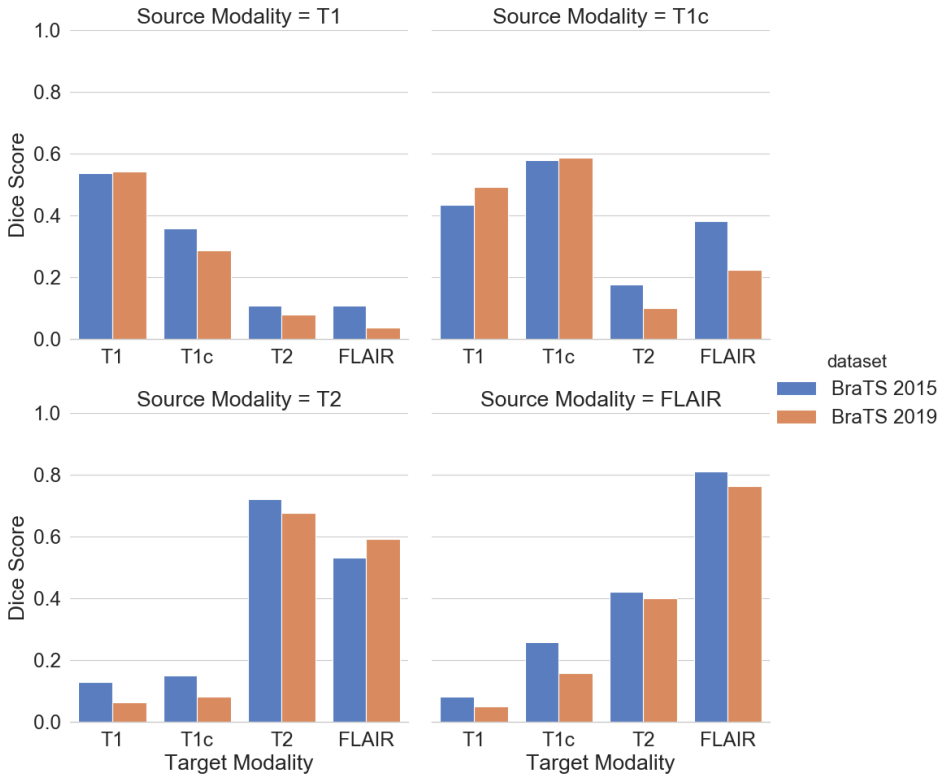


Figure 6.11: *Performance of each uni-modal model when applied to images acquired using other modalities. The performances on both BraTS 2015 and BraTS 2019 are reported. In addition, we also report as a reference the performance of the uni-modal models on images with the same modality the models are trained for.*

Target	Fine Tuning	BraTS 2015			BraTS 2019		
		Dice Score	Precision	Sensitivity	Dice Score	Precision	Sensitivity
T1	S,D	0.496	0.503	0.553	0.502	0.571	0.582
T1	S,D_{in}	**0.561**	0.614	0.576	0.528	0.558	0.538
T1c	S,D	0.467	0.464	0.538	0.468	0.527	0.494
T1c	S,D_{in}	0.577	0.661	0.541	**0.598**	0.705	0.563
T2	S,D	0.692	0.776	0.668	0.674	0.643	0.775
T2	S,D_{in}	**0.781**	0.818	0.771	**0.741**	0.878	0.681

Table 6.17: *Performance of the models trained by transferring the model trained on FLAIR images. The column* `Fine Tuning` *reports whether both the model networks (*`S,D`*) or only the segmentator and the discriminator input layer (*`S,`D_{in}*) have been trained on the target modality. The results are reported for each target modality as the average of the dice score, the precision, and the sensitivity computed on each one of the MRI volume included in test set of BraTS 2015 and BraTS 2019. We reported in bold the scores that are better than the corresponding ones in Table 6.16.*

The same behavior, the other way around, is found in the performance of the models trained on T2 and FLAIR.

Based on the results discussed above, we applied the transfer learning to train three uni-modal models respectively on T1, T1c, and T2 images by adapting a model trained on FLAIR images. Our results show that the FLAIR images account for a large part of the model performance, and our purpose is to understand whether the transfer learning process might improve the final performances. In this experiment, the fine-tuning of the model on the target modality was limited to 300 epochs.

Table 6.17 shows the performance of the models trained on T1, T1c, and T2 by adapting a model trained on FLAIR. The results suggest that it is more convenient to adapt, i.e., to train on images with the target contrast modality, *only* the input layer of the discriminator network of the model, keeping the other layers of the discriminator network trained on images with the FLAIR contrast modality. We believe that this result is due to the high instability of the adversarial learning mechanism that makes it very difficult to adapt a previously trained model [225] incrementally. Accordingly, to exploit the knowledge of the model trained on images with FLAIR contrast, we adapted on the target modalities only the segmentation network and the first layer of the discriminator network. Overall, this solution often led to models that perform better or similar to the models trained from scratch on the target modalities (see Table 6.16), despite being trained for 300 epochs only. As expected, the significant benefits of the transfer learning mechanism are achieved on the target modality more similar to FLAIR, i.e., T2.

6.4 Addressing Data Heterogeneity using Generative Adversarial Networks

Many different approaches and datasets have been used in the literature to overcome the problem of missing modalities. Since most datasets only contain T_1 or T_1/T_2/PD scans due to practical reasons, Orbes-Arteaga et al., in [68], implemented a GAN that generates Brain FLAIR using the T_1 modality. Camileri et al. [239] developed a variant of the original GAN, called Laplacian pyramid framework (LAPGAN), that synthesizes images in a coarse-to-fine process by introducing progressive refinements. As the name suggests, this method is based on Laplacian pyramid and allows to initially generate an image with low resolution and then incrementally refine it by adding finer details. Another approach to generate missing modalities was proposed in [66] where the authors presented two possible scenarios, based on the given dataset: (i) when the multi-contrast images are spatially registered they use a model called pGAN, which incorporates a pixel-wise loss into the objective function, while they adopt (ii) a cycleGAN [65] in the more realistic scenario in which pixels are not aligned between modalities. As we introduced in previous chapters, Anmol Sharma and Ghassan Hamarneh [64] proposed a *many to many* generative model, capable of synthesizing multiple missing sequences given a combination of various input sequences. Furthermore, they also apply the concept of curriculum learning based on the variation of the difficulty of the examples used during the network training.

Finally, GANs have been successfully applied also to different images, such as PET, CT, and MRA images. Notable examples include the work of Olut et al. [240], which proved that GANs work efficiently even when the source imaging technique is different from the target one by synthesizing MRA brain images from T_1 and T_2 MRI modalities. Then, Ben-Cohen et al.[241] successfully generated PET images using CT scans through a fully connected neural network, whose output is improved and refined by cGANs.

6.4.1 Modality Generation with Adversarial Networks

Focusing on the issue of missing MRI modalities, we studied the problem of generating a target brain MRI modality among the four available in the BraTS dataset: T1, T1ce, T2, FLAIR. In particular, we assumed that a missing modality has to be generated by the other three, which are supposed to be available. To this purpose, we focused on multi-input gen-

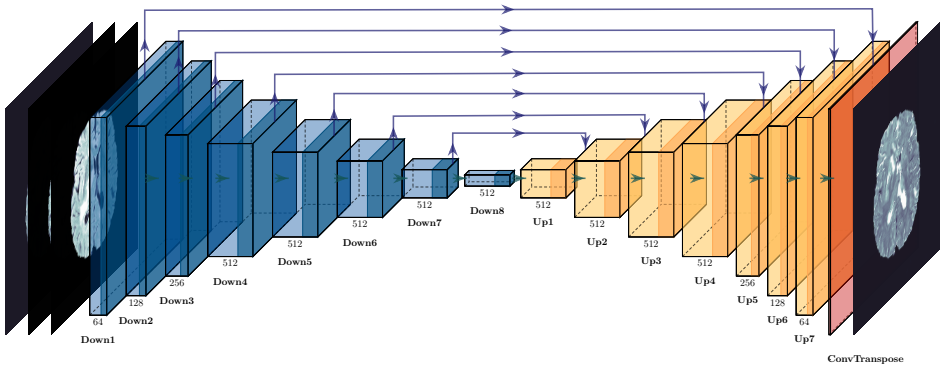
erative models, i.e., models that receive multiple images as input – i.e., the available modalities– and generate as output a single image –i.e., the missing modality–. In particular, we designed two generative models based on GANs: a multi-input version of pix2pix[63], dubbed *MI-pix2pix*, and a modified version of the MM-GAN introduced by Sharma et al.[64], dubbed *MI-GAN*. In this section, we provide the details of these two generative models.

6.4.2 MI-pix2pix

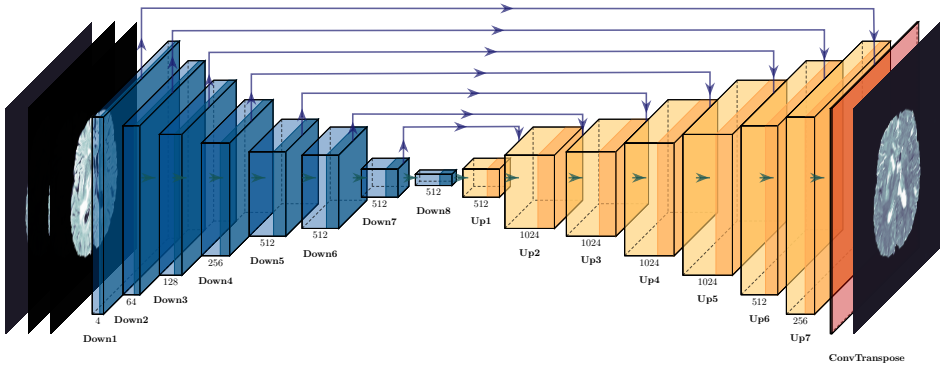
In this GAN, the generator is based on U-net [179] that takes as input a tensor with size 32x256x256x3, where the first dimension indicates the batch size and the last one is the number of modalities in input. For example, when MI-pix2pix is used to generate the missing T_2 modality, the other three modalities, T_1, T_{1c}, T_{flair}, are provided as inputs respectively as the first, second and third channel. The symmetrical downsampling and upsampling blocks, typical of U-Net architecture, employ skip connections to concatenate the inputs of each downsampling block to the input of the corresponding upsampling block. The downsampling blocks are composed by three layers: $C_{n,k=4,s=2}$, BatchNorm, LeakyReLU, where $C_{n,k,s}$ is a convolutional layer with n filters, kernel size k and stride s. The sequence of downsampling blocks, with n={64,128,256,512,512,512,512,512}, reduces the spatial information and increases the feature dimension until the last downsampling block that has an output shape of 1x1x512. Instead, the upsampling blocks are composed by four layers: $D_{n,k=4,s=2}$, BatchNorm, Dropout, ReLU, where $D_{n,k,s}$ is a transposed convolution layer with n filters, kernel size k and stride s. The sequence of upsampling blocks, with n={512,512,512,512,512,256,128,64}, is followed by a last transposed convolutional layer and a *Tanh* activation. The output of the network is a batch of images with dimension 256x256x1. The discriminator is a 70x70 PatchGAN with two inputs: (i) the target image (fake or real) of shape 32x256x256x1, (ii) the concatenation of the three modalities (of shape 32x256x256x3) provided as input to the generator. This network has only three downsampling blocks (with $n = \{64, 128, 256\}$), followed by these layers: $ZeroPadding$, $C_{n=512,k=4,s=1}$, $BatchNorm$, $LeakyReLU$, $ZeroPadding$, $C_{n=1,k=4,s=1}$.

6.4.3 MI-GAN

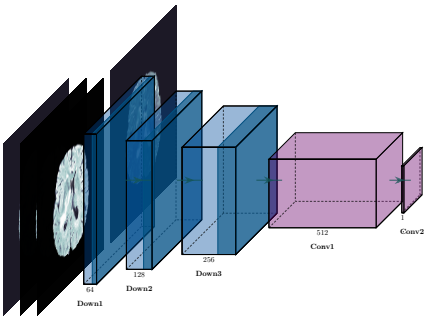
The MI-GAN architecture is based on the MM-GAN [64]: the only variation we applied was the replacement of every layer of Instance Normal-



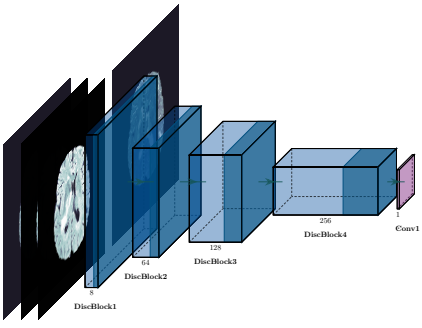
(a) *MI-Pix2Pix Generator*



(b) *MI-GAN Generator*



(c) *MI-Pix2Pix Discriminator*



(d) *MI-GAN Discriminator*

Figure 6.12: *Generator and Discriminator architectures of the two multi-input GANs studied in this work.*

ization with a Batch Normalization, which is also used in MI-pix2pix, as we observed that normalizing the activations of each channel across the whole batch was more effective, in terms of quality in the generated samples, than computing the mean/standard deviation and normalizing across each channel for each training image. The MI-GAN generator is a modified U-Net with concatenated skip connections and the typical U-shape architecture (Figure 6.12b), The building blocks for the encoding path are defined as: $C_{n,k=4,2}$, $BatchNorm$, $LeakyReLU$, $DropOut_{0.5}$. The *upsample block* has the same layers seen in the pix2pix architecture: $D_{n,k=4,s=2}$, $BatchNorm$, $ReLU$, $Dropout_{0.5}$ The last layer of the generator is a transposed convolution followed by a *Tanh* activation function. The output of the network is a batch of images with dimensions 256x256x1. The discriminator takes two inputs X_t and X_i and produces one output D(X_t, X_i), each one of these with 4 channels - i.e. one per modality. Assuming the discrimination of a T_1' synthesized scan, the inputs were X_t: {T_1, T_2, T_{1c}, FLAIR} and X_i: {T_1', T_2, T_{1c}, FLAIR}.

6.4.4 Loss Function

The generator loss and discriminator loss used with MI-GAN (6.4.3) are the ones proposed in [63] and defined as follows:

$$\begin{aligned} L_G &\leftarrow \lambda \mathcal{L}_1(G(x), y) + (1-\lambda)\mathcal{L}_2(D(x, G(x)), L_{ar}) \\ L_D &\leftarrow \mathcal{L}_2(D(x, y), L_{ar}) + \mathcal{L}_2(D(x, G(x)), L_r) \end{aligned} \tag{6.4}$$

where the input x is a concatenation of three sequences, while $G(x)$ represents the prediction generated by the GAN. L_{ar} is a tensor of unitary values that is used to encourage the generator to produce samples that the discriminator evaluates as real. $\mathcal{L}_2$ is the L2 norm - or *Mean Squared Error* -, while $\mathcal{L}_1$ denotes the L1 norm - or *Mean Absolute Error* that is useful to generate samples that are structurally similar to the target image. This term was chosen as a reconstruction loss term because of its ability to prevent blurring compared to using an L2 loss [64]. L_r is a tensor with entries equal to zero, and it is used to encourage D to assign low values to the generated samples. In the same way, L_{ar} is used to induce the discriminator in assigning values close to 1 to the true samples.

Following the same notation, the loss of MI-pix2pix is instead defined as:

$$\begin{aligned} L_G &\leftarrow \lambda \mathcal{L}_1(G(x), y) + BCE(L_{ar}, D(x, G(x))) \\ L_D &\leftarrow BCE(D(x, y), L_{ar}) + BCE(D(x, G(x)), L_r) \end{aligned} \tag{6.5}$$

where $BCE(x, y)$ is the *Binary Cross-Entropy* as commonly implemented in most deep learning frameworks. The L_D of MI-pix2pix is computed as the sum between the binary crossentropy of $(D(x, y), L_{ar})$ and the one of $D((x, y'), L_r)$ where y' is the generated image. L_G, on the other hand, contains the reconstruction term between y and y', multiplied by an hyper-parameter λ and summed to the binary cross-entropy of $D((x, y'), L_r)$.

6.4.5 Experimental Design

To test our approach, we used data from the BraTS 2015 dataset. We split the dataset into three different sets: training (80%), validation (10%), and test (10%), resulting in 219 patients assigned to the first set, 27 to the validation one, and 28 to the test. Since we believe it is important to maintain the balance between HG and LG subjects during training and evaluation phases, we split the dataset by applying stratified sampling [185]. All the images in the dataset were first *center cropped*: the outer parts of each volume were removed while the central region was retained along each dimension. We also discarded the external slices that contained almost no useful information, reducing the number of slices from 155 to 128. As a result, the final shape of each volume is 180x180x128. Then, we applied a min-max normalization to each volume to provide data with the same dynamic range to the models. Finally, as our GANs architecture only allows input images with dimensions that are a power of 2, we padded our 180x180 images to obtain 256x256 images.

Evaluation Metrics

To evaluate the output of our generative models, we considered different metrics that could be used to assess three different objectives: (i) quality of the whole image, (ii) quality of the tumor area, and (iii) discriminative power of the generated image.

Image Metrics. The first metrics we considered aim at assessing the quality of the whole images generated. To compute these metrics, we first had to crop and normalize the images as follows. We center-cropped the generated images to 155x194, which is the size of the largest bounding box to contain each brain in the BraTS2015 dataset [185]. Then, we applied *mean normalization* [242] to each image, either generated or real. Thus, we computed the following three metrics: (i) the mean squared error (MSE), (ii) the *Peak Signal-to-Noise Ratio* (PSNR) [199], and (iii) the *Structural Similarity* (SSIM) [200], already described in Chapter 3.

Tumor Metrics. Since the dataset we considered aims to perform tumor segmentation, we also considered the performance in this area in our analysis. To this extent, we also computed the MSE and PSNR metrics by restricting the computation only to the pixels inside the image's tumor area.

Discriminative Metrics. In addition to the metrics described so far, we compared the performance achieved by a tumor segmentation model when generated images are used as input instead of real ones to assess the amount of information contained in the generated images. More specifically, we computed the *Dice Similarity Coefficient* of the segmentations obtained from the SegAN-CAT model, introduced in Section 6.3.

6.4.6 Results

We compared MI-GAN and MI-pix2pix on the generation of each one of the four MRIs modality available in the BraTS2015 dataset: T_1, T_{1c}, T_2, and T_{2flair}. We also compared the performance of MI-GAN and MI-pix2pix with some single-input pix2pix generative models. As a performance baseline, we computed the metrics of the images provided as input to the generative models –i.e., we evaluated how similar the input images are to the expected output of the generative models.

To train the pix2pix, MI-pix2pix and MI-GAN, we used the following parameters settings. The learning rate α was set to 0.0002, the exponential decay rate for the 1st moment estimates β_1 was set to 0.5, and the one for the 2nd moment estimates (β_2) was set to 0.999. The value of λ was set to 100 in the generator loss of pix2pix and MI-pix2pix, while it was set to 0.9 in the loss of the MI-GAN generator. The MI-GAN discriminator loss was multiplied by 0.5 to slow down the rate at which D learns compared to G. The convolutional layer weights have been initialized using a normal distribution with 0 mean and standard deviation equal to 0.05. The training was performed for 25 to 70 epochs until convergence of each model. In the results reported in this section, we used the following notation. The pix2pix models have been dubbed as P2P and MI-pix2pix as MI-P2P. In addition, for pix2pix and the baseline performance, we reported the source modality used between parenthesis.

Generation of T_1 images. We trained four models to generate the missing modality T_1: two single-input pix2pix with different inputs (T_2 and T_{1c}, MI-pix2pix, and MI-GAN. We choose to train different pix2pix models to understand how the performances change when using an input, T_{1c}, that is similar to the target compared to using T_2. The results are reported in Table 6.18.

Model	MSE	PSNR	SSIM	$\text{MSE}_{\text{tumor}}$	$\text{PSNR}_{\text{tumor}}$
Baseline(T_2)	0.0396 ± 0.0275	15.3286 ± 4.2134	0.5054 ± 0.2116	0.0594 ± 0.0523	13.6678 ± 3.6085
P2P(T_2)	0.0060 ± 0.0046	23.4967 ± 3.6754	0.8112 ± 0.1004	0.0199 ± 0.0187	18.5047 ± 3.7607
Baseline(T_{1c})	0.0058 ± 0.0050	23.8431 ± 4.0912	0.8096 ± 0.0984	0.0173 ± 0.0216	20.1544 ± 5.0543
P2P(T_{1c})	0.0044 ± 0.0041	25.0680 ± 3.8652	0.8403 ± 0.0856	0.0114 ± 0.0143	21.4485 ± 4.3380
MI−P2P	0.0044 ± 0.0040	24.9339 ± 3.6983	0.8413 ± 0.0838	0.0113 ± 0.0099	20.8938 ± 3.6111
MI−GAN	$\mathbf{0.0041 \pm 0.0038}$	$\mathbf{25.2569 \pm 3.6512}$	$\mathbf{0.8472 \pm 0.0830}$	$\mathbf{0.0102 \pm 0.0097}$	$\mathbf{21.5359 \pm 3.8620}$

Table 6.18: *Generation of T_1: performances on the test set. Best results are reported in bold.*

Model	MSE	PSNR	SSIM	$\text{MSE}_{\text{tumor}}$	$\text{PSNR}_{\text{tumor}}$
Baseline(T_1)	0.0396 ± 0.0275	15.3286 ± 4.2134	0.5054 ± 0.2116	0.0594 ± 0.0523	13.6678 ± 3.6085
P2P(T_1)	0.0100 ± 0.0074	21.3182 ± 3.8023	0.7521 ± 0.1247	0.0476 ± 0.0397	14.3652 ± 3.3523
Baseline($T_{2\text{flair}}$)	0.0275 ± 0.0199	16.8268 ± 3.9727	0.6262 ± 0.1597	0.0464 ± 0.0500	15.1591 ± 4.0428
P2P($T_{2\text{flair}}$)	0.0087 ± 0.0076	21.9227 ± 3.7021	0.7567 ± 0.1287	0.0256 ± 0.0242	17.3035 ± 3.4584
MI−P2P	$\mathbf{0.0073 \pm 0.0063}$	$\mathbf{22.7645 \pm 3.8272}$	$\mathbf{0.8005 \pm 0.1112}$	0.0207 ± 0.0167	$\mathbf{18.1305 \pm 3.5930}$
MI−GAN	0.0077 ± 0.0061	22.3719 ± 3.5290	0.7835 ± 0.1141	$\mathbf{0.0205 \pm 0.0167}$	18.0725 ± 3.3763

Table 6.19: *Generation of T_2: performances on the test set. Best results are reported in bold.*

Generation of T_2 images. For the generation of T_2 we trained four models: the first one is a pix2pix trained to receive T_1 as input, the second one takes as input the T_{2flair} modality, that captures more similar characteristics (of T_2) than T_1; the other models are MI-pix2pix and MI-GAN. The results are reported in Table 6.19.

Generation of T_{1c} images. Table 6.20 shows the performances obtained by three models trained to generate T_{1c} slices: one pix2pix model and two multi-input models. We choose T_1 as the input modality for the single-input GAN because it is the most similar sequence, among the ones available, to the target T_{1c}.

Generation of T_{2flair} images. Tables 6.21 and 6.22 summarize the performances reached by the models trained to generate T_{2flair}: in particular 6.22 shows the additional metric we implemented to evaluate the quality of the generated tumor area using the SegAN-CAT model. The DSC is calculated between the ground truth and the segmentation of our synthesized images. Furthermore, a reference DSC score is computed between the ground truth

Model	MSE	PSNR	SSIM	$\text{MSE}_{\text{tumor}}$	$\text{PSNR}_{\text{tumor}}$
Baseline(T_1)	0.0058 ± 0.0050	23.8431 ± 4.0912	0.8096 ± 0.0984	0.0173 ± 0.0216	20.1544 ± 5.0543
P2P(T_1)	$\mathbf{0.0051 \pm 0.0048}$	$\mathbf{24.6165 \pm 4.0755}$	$\mathbf{0.8139 \pm 0.0996}$	$\mathbf{0.0155 \pm 0.0199}$	$\mathbf{20.6981 \pm 4.9762}$
MI−P2P	0.0052 ± 0.0040	24.1597 ± 3.8631	0.8110 ± 0.0963	0.0168 ± 0.0172	19.8441 ± 4.6258
MI−GAN	0.0054 ± 0.0040	23.9242 ± 3.6958	0.8027 ± 0.1003	0.0157 ± 0.0162	19.9779 ± 4.3568

Table 6.20: *Generation of T_{1c}: performances on the test set. Best results are reported in bold.*

Model	MSE	PSNR	SSIM	$\mathbf{MSE_{tumor}}$	$\mathbf{PSNR_{tumor}}$
Baseline(T_2)	0.0275 ± 0.0199	16.8268 ± 3.9727	0.6262 ± 0.1597	0.0464 ± 0.0500	15.1591 ± 4.0428
P2P (T_2)	0.0090 ± 0.0065	21.5895 ± 3.4831	0.7518 ± 0.1211	0.0390 ± 0.0463	15.9946 ± 4.0459
MI−P2P	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8165 \pm 3.7317}$	$\mathbf{0.7772 \pm 0.1094}$	$\mathbf{0.0221 \pm 0.0375}$	$\mathbf{19.0374 \pm 4.1582}$
MI−GAN	0.0072 ± 0.0050	22.5524 ± 3.5655	0.7610 ± 0.1175	0.0258 ± 0.0285	17.4694 ± 3.6137

Table 6.21: *Generation of T_{2flair}: performances on the test set. Best results are reported in bold.*

and the segmentation of the original slice from BraTS2015. We trained only one pix2pix model, using as input T_2 that is the most similar sequence, among the ones available, to the target.

Table 6.22: *DSC performances, on the test set, obtained by comparing the ground truth and the segmentations, using SegAN-CAT, of T_{2flair} predictions.*

Segmentation image	$\mathbf{DSC_{tumor}}$
Original T_{2flair}	0.8053 ± 0.1156
P2P(T_2)	0.6632 ± 0.1608
MI-P2P	$\mathbf{0.7427 \pm 0.1810}$
MI-GAN	0.6837 ± 0.2136

6.4.7 Modality Replacement on Multi-Input SegAN-CAT

In the last experiment, we wanted to complete our analysis by investigating whether the generated modalities can be used for segmentation in place of the missing modalities. For this experiment, we considered the *multi-modal* SegAN-CAT, trained on BraTS 2015. For each of the four modalities, we tested the performances of SegAN-CAT by replacing one of the modalities with the ones generated by either MI-Pix2Pix or MI-GAN, using the other three modalities as input for the generation. Results are reported in Table 6.23.

6.4.8 Discussion

In this section, we discuss the result reported in the previous section. In the first part of the discussion, we discuss the quantitative results we obtained. In the last part, we will provide a brief qualitative analysis of the generated images.

6.4.9 Quantitative Analysis

Pix2pix vs Baselines. Our results (see Table 6.18, Table 6.19, Table 6.20, and Table 6.21) show that, as expected, pix2pix is indeed able to generate images that are more similar to the images of the target modality with

Model	Generated Modality	DSC	Precision	Sensitivity
SegAN-CAT (baseline)	-	*0.8588*	*0.8820*	*0.8519*
MI-GAN	T1	0.7385	0.7487	0.7748
MI-pix2pix	T1	0.6366	0.5880	0.7805
MI-GAN	T1c	0.8549	0.8879	0.8401
MI-pix2pix	T1c	0.8548	0.8867	0.8406
MI-GAN	T2	0.8278	0.8807	0.7971
MI-pix2pix	T2	0.8321	0.8861	0.8000
MI-GAN	Flair	0.7483	0.8499	0.7007
MI-pix2pix	Flair	0.3545	0.7902	0.2529

Table 6.23: *Performances of our Image translation models applied to a multi-modal SegAN-CAT. For each modality in the BraTS 2015 test set, we replaced the corresponding images with the samples generated by MI-GAN and MI-P2P using the other three modalities and evaluated the segmentation model performances. The* baseline *row indicates the original performances of SegAN-CAT without image replacement.*

respect to the ones received as input. All the metrics computed on the baselines are worse than the corresponding ones computed on the images generated with pix2pix generative models. As an example, when pix2pix is trained to generate T_{2flair} images from T_2 images (see P2P(T_2) in Table 6.21), it is able to achieve improvements over the baseline (i.e., T_2 images) of 35%, of 38%, and of 560% respectively in PSNR, SSIM, and MSE. These results suggest that the processing performed by the generative models can *translate* the input into images that are more similar to the target ones.

Single-Input vs Multi-Input Models. Our results show that both multi-input models generally outperform single-input ones. However, our results show at least two cases where a single-input model can perform better than a multi-input one. In particular, the results in Table 6.18 show that, when trained to generate T_{1c} images from T_1, single-input pix2pix is able to outperform MI-pix2pix (but is outperformed by MI-GAN). Instead, Table 6.20 shows that the single-input pix2pix trained to generate T_{1c} from T_1 images outperforms both MI-pix2pix and MI-GAN on this task. These results are not very surprising as T_1 and T_{1c} are very similar modalities, and the information content of T_2 and T_{2flair} modalities do not provide a clear advantage to generate the target images. Therefore, in this kind of task, a simpler single-input model can perform better than a more complex one. These results suggest that the choice of single-input models over multi-input ones should be based on a careful analysis of the considered task.

MI-pix2pix vs MI-GAN. Our results show that MI-pix2pix performs slightly better than MI-GAN for the generation of all the modalities except for T_1, where MI-GAN achieves the best performance instead. The results also suggest that the improved performances of MI-GAN compared to pix2pix discussed in [64] may not be due to the novel architecture and loss function of MI-GAN but instead to the benefit of using multiple modalities as input. As our results show, when pix2pix is extended to use multiple modalities as input (MI-pix2pix), it generally outperforms MI-GAN.

Whole image vs tumor area. Comparing the metrics computed on the whole image to the ones computed on the tumor area shows that none of the trained models seems to perform much better (or much worse) than the others on the specific tumor area. On the other hand, a direct comparison of the values computed on the tumor area with the values computed on the whole image is not possible. While for the tumor area, the computation involves only relevant pixels, for the whole image, it also involves pixels with no information (black pixels). Therefore, further investigations will be necessary to understand this issue better.

Discriminative Metrics. We also tried to assess the discriminative power of the generated images when used to make some decisions, more specifically, when used as input to a single-input segmentation model. We performed this analysis on T_{2flair} images, as they are the most effective ones in segmentation tasks. The results in Table 6.22 show that the images generated with MI-pix2pix are the ones that allow achieving the best segmentations, reaching an average performance (the DSC score) that is rather good compared to the ones reached using the real images.

Modality Replacement on Multi-Input SegAN-CAT Lastly, we tried to assess how the performances of a Multi-Input segmentation model change when an unavailable modality in input is replaced with images generated starting from the other three. To this extent, we applied our image translation models to the BraTS 2015 dataset and tested the performances of our multi-modal SegAN-CAT adversarial network. The results show that in particular, for the T_{1c} and T_2, the performances are almost the same as those obtained using real images. However, the MI-pix2pix approach fails, in this case, to provide Flair samples useful for the multi-input SegAN-CAT, while MI-GAN accomplishes the task with a modest loss in final performances. We envision that this result might be due to the feature extraction process used by SegAN-CAT, which might penalize some features present in the images generated by MI-pix2pix. This claim is also supported by the fact that our previous results already asserted the validity of the MI-pix2pix

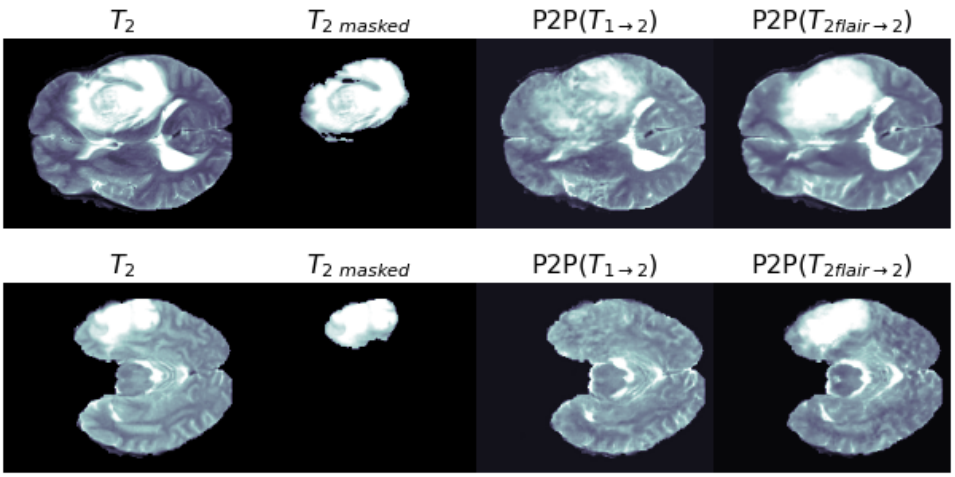


Figure 6.13: *Comparison of P2P($T_{1\to 2}$) (i.e., pix2pix trained to generate T_2 from T_1) and P2P($T_{2flair\to 2}$) (i.e., pix2pix trained to generate T_2 from T_1). $T_{2masked}$ is the tumor area.*

when used with a single-input SegAN-CAT model, although indicating that further investigations may provide helpful insights on the problem.

6.4.10 Qualitative Analysis

The qualitative analysis of the generated images allows to confirm the findings discussed above. Figure 6.13 shows two examples of T_2 images generated by pix2pix using either T_1 or T_{2flair} as input. A qualitative analysis shows how using as input a modality that is more similar to the target one, i.e., T_{2flair} instead of T_1, allows to generate much better quality images, especially in the tumor area.

On the other hand, a qualitative analysis of this kind does not always allow to appreciate the differences among the models. As an example, Figure 6.14 shows some T_2 images generated by different models.

Finally, Figure 6.15 shows some examples of segmentation obtained using either real or generated images. As expected, the segmentations obtained from generated images are generally not as good as those obtained from real images. However, sometimes the segmentation obtained from generated images are very similar (e.g., fourth row in Figure 6.15) or even better (e.g., second row in Figure 6.15) than the ones obtained from the real images. This might also be due to the fact that generated images include information from other modalities that might be not present in the real image.

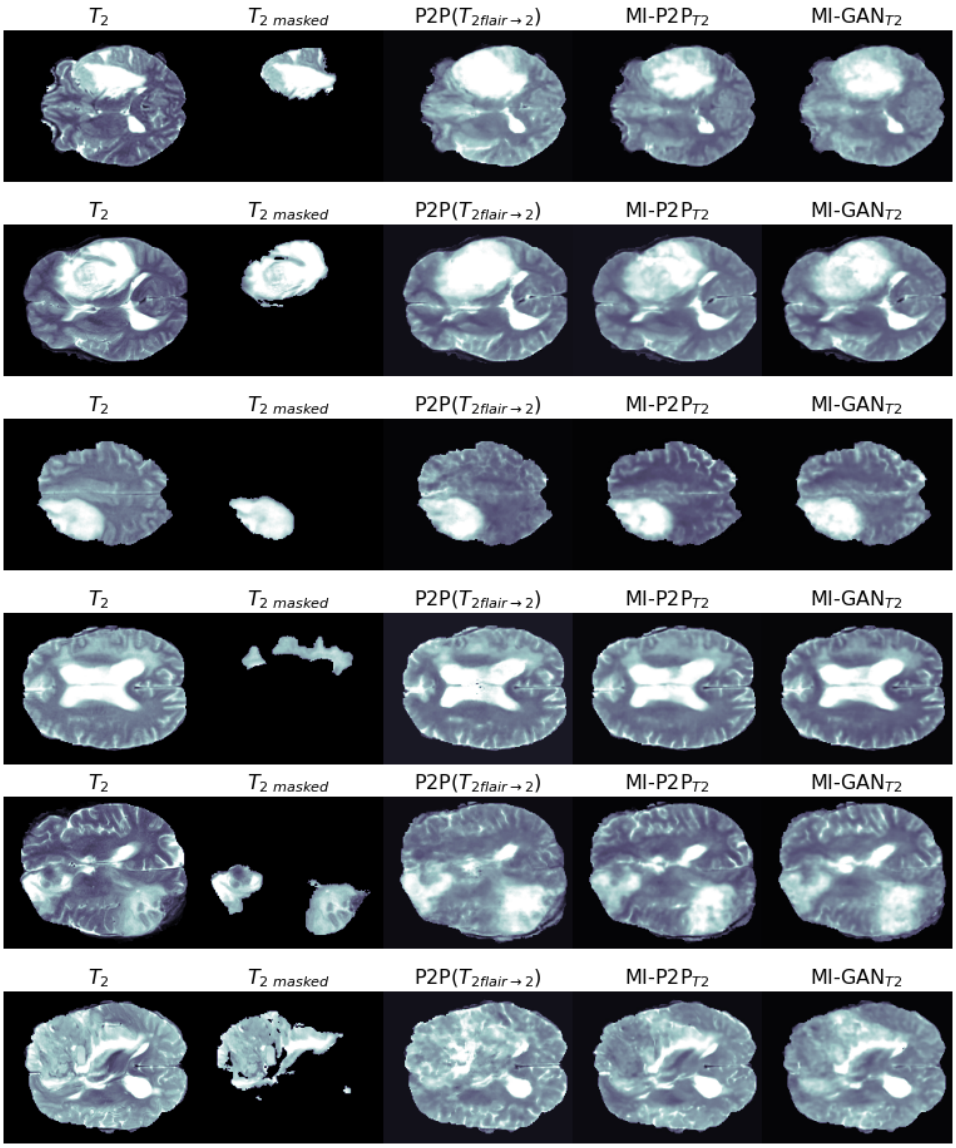


Figure 6.14: *Some examples of real and generated T_2 images. P2P($T_{2flair\rightarrow 2}$) is a single-input pix2pix model that uses T_{2flair} images as input, MI-P2P$_{T_2}$ and MI-GAN$_{T_2}$ are multi-input models trained to generate T_2 images. Each row corresponds to a different subject from the test set.*

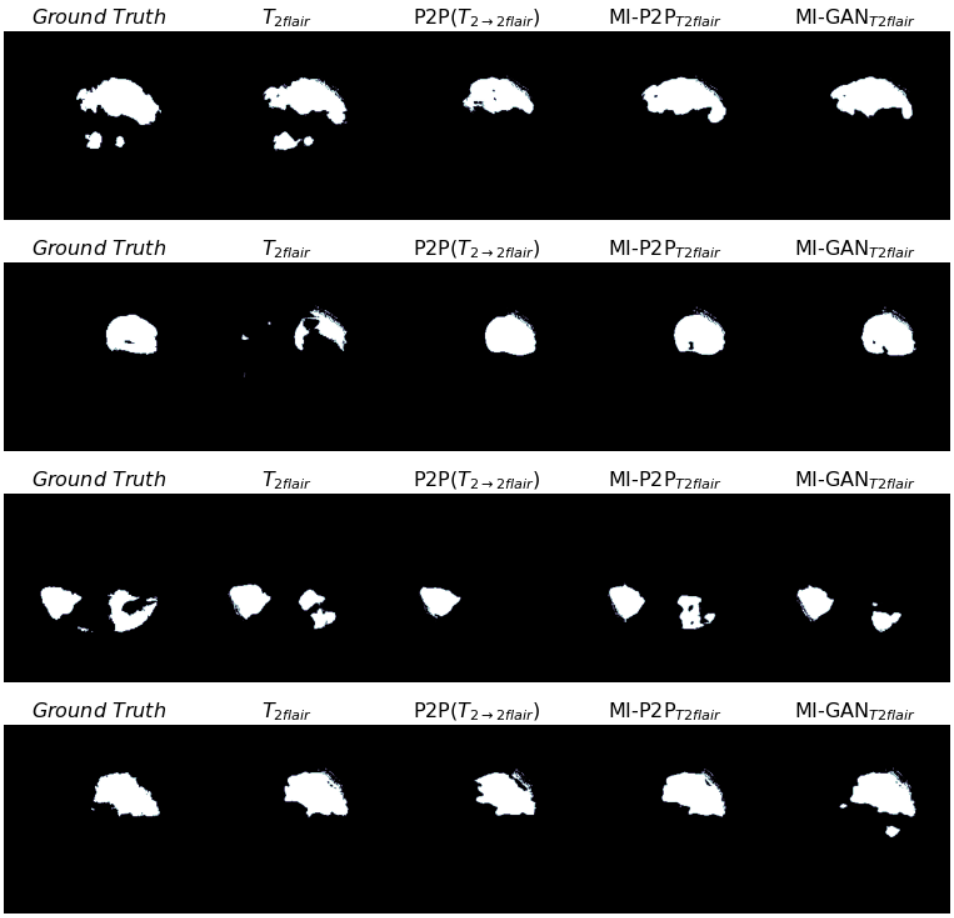


Figure 6.15: *From left to right: ground truth and the segmentations from T_{2flair}, pix2pix, MI-pix2pix and MI-GAN. Each row corresponds to a different subject from the test set.*

6.5 Summary

In this chapter, we used transfer learning to address different issues in collaborative learning applied to medical imaging. The first was related to the need for an automated feature extraction pipeline that also allowed to train more specialized models for disease classification, which could potentially be more suitable for smaller private datasets.

For this reason, we proposed an approach for training different kinds of machine learning models –i.e., not only neural networks–, starting from a pre-trained CNN. In particular, our first goal was to assess whether the embeddings of Chest X-Rays extracted from a CNN might be used to train novel classifiers from scratch.

To this purpose, we extracted the image embeddings from our CNNs trained on CheXpert and used them to train two sets of classifiers based on Random Forest and eXtreme Gradient Boosting (XGBoost). Then, we completed the experiments by applying our embedding procedures to obtain the final scores on CheXpert, using the combination of all the models at our disposal. Our results show that image embeddings retain enough relevant information to train classifiers based on trees. Moreover, the performances obtained are often even better than those achieved by the NN classifiers, indicating that CNNs are optimal for model extraction. However, the classifier layer may require large amounts of data to achieve the best performances. Once more, ensembling has proven to be useful in combining classifiers, in particular when no single classifier outperforms the others on all the target labels. Regarding the ensembling strategy, the entropy-weighted averaging allowed again to achieve better performance overall by assigning more weight to the most confident classifier for each label.

Once we expanded our model collection on CheXpert, we explored the role of transfer learning and embedding for domain and task adaptation in medical imaging. In particular, we exploited our pre-trained models to learn a set of models on a smaller private dataset with a different label than CheXpert. The advantage of using embeddings is exploiting the feature extraction capabilities of CNNs trained on a large dataset of images while designing a specific classifier for new data and, eventually, for a slightly different task.

To understand the best approach to the problem, we proposed two different transfer learning approaches to apply the feature extractor stage of pre-trained CNNs to a new local independent dataset - HUM-CXRs. Our first TL approach consisted in applying a conventional fine-tuning of the last classification layer of each CNN. This approach allowed to adapt the

vector of labels to the new dataset and update the classifier's weights to the new task. The second approach involved two steps: First, we extracted the image embeddings from the CNN, then we trained a set of tree-based models to classify the embeddings. The latter approach has the dual advantage of providing the best classification performances and minimizing the computational resources – and time – required for adapting the pre-trained models to the new task. This approach is thus successful when computational power – such as GPU availability – is limited.

The model pre-trained on CheXpert performed poorly on the baseline approach (best average AUROC 0.777), i.e., using the CNNs directly in inference on the new external independent dataset. Therefore, even if they were trained on an extensive dataset, they could not generalize to a new domain and additional data. On the other hand, using transfer learning, in particular with image embeddings, it was possible to adapt the original models to a new domain, –i.e., new hospital, new geographic and demographic characteristics– and new tasks –i.e., different labels–, with minimum effort and competitive performance (best average AUROC 0.856). Our approach is not limited to our dataset and application; it could be adopted and successfully applied by any other research group or hospital that might need to classify medical images but does not have either a sufficient volume of data or the computational resources to train the model. The resulting models would have an excellent feature extraction capability learned from large public datasets following this framework. However, they will be validated, tailored, and improved to the specific application to achieve optimal results. Another relevant advantage is that neither raw data nor other related information would need to leave the hospital that owns it.

We used a Grad-CAM approach to provide an explainable insight into our final model and investigate the presence of spurious correlations and dataset biases. Overall, the explanations provided by the algorithm showed a good ability to identify specific features concerning the identified classes. The produced saliency maps focus on the lung field, with particular attention to the correct side of the chest. Double class images correctly showed the differences between chest findings. However, De Grave et al. [81] claimed to be skeptical about presenting only a few examples of explanations since they may not truthfully represent the real behavior of the model. For this reason, we also presented population-level explanations by averaging two batches of 200 CXRs. The averaged saliency maps report a correct focus of the network on the center of the image without exploiting shortcuts that could be present outside the lung field, such as annotations, bor-

ders, and lateral markers. The only exception we have identified concerns the device class, particularly when detecting a pacemaker. While in some cases it correctly focuses on the hardware components, in some examples (Figure 6.9) it correctly classifies "device", but it exploits the leads that from the pacemaker goes inside the atrium of the heart. This finding is not entirely wrong, but we would expect it to focus more on the hardware – i.e., the main box–. We think this might be caused by the original dataset on which the models were pre-trained. The device class is extensive and includes lines, tube, valve, catheter, pacemaker, hardware, coil, etc. Unfortunately, the percentage of each subclass is not public. However, it might be possible that tubes, leads, electrodes, and catheters are more present than pacemakers, inducing the model to focus on them. Furthermore, still concerning device class, we investigated false positive predictions. In most cases, we asserted that the model correctly classified the device while the ground truth was wrong. The main reason behind these false positives is that our labels were extracted from the medical report that is often unstructured and operator-dependent. While pathologies are written and discussed in the report, pacemakers, electrodes, probes, and others are sometimes not clearly described in the report because not considered as "abnormal" as medical pathologies.

The second issue we investigated in this chapter is missing modalities in medical imaging. As we already introduced in chapter 2, medical imaging datasets acquired in clinical practice are often incomplete, and this could limit the applicability of models that instead require multiple modalities as input. To investigate this issue, we first introduced SegAN-CAT, an adversarial network architecture based on *SegAN* [185]. Our approach differs from *SegAN* mainly in two respects: (i) the loss function has been extended with a dice loss term and (ii) the input of the discriminator network consists of a concatenation of the MRI images and their segmentation.

We applied SegAN-CAT to the Brain Tumor Segmentation problem, the same task the *SegAN* architecture was successfully applied to. We then designed a set of experiments to (i) compare the performance of SegAN-CAT and *SegAN*, (ii) assess the performance of uni-modal models for each contrast modality (i.e., T1, T1c, T2, and FLAIR), (iii) to study the problem of transferring a previously trained model across different modalities. To this purpose, we first trained *SegAN*, *SegAN* with a dice loss term, and SegAN-CAT on MRIs acquired with all the four contrast modalities. Our results on both BraTS 2015 and BraTS 2019 datasets showed that both the dice loss term and the discriminator input concatenation proposed in this experiment improve the model's final performance. Then, we trained one

uni-modal SegAN-CAT model for each one of the four contrast modalities in order to assess the performance that can be achieved using only the information available in each modality alone. As expected, none of these four models reached the performance of the multi-modal one. However, the model trained on the FLAIR contrast modality outperformed all the others and reached a performance close to the one achieved by the multi-modal model. Thus, we investigated the possibility of transferring a model trained with FLAIR to train a better model from images acquired with different contrast modalities. Despite confirming that it is rather difficult to use transfer learning with adversarial networks [225, 227], our results suggest that it is possible to successfully transfer a model trained on FLAIR contrast modality also to other modalities. Indeed, our results show that transfer learning often allows training uni-modal models with better performance than the same models trained entirely on images with their specific contrast modality.

In our last set of experiments, we investigated the possibility of generating an MRI modality from the available ones, exploiting image translation. To this extent, we developed and compared different generative models based on GANs to produce synthesized MRI modalities. In particular, we studied two different multi-input generative models. The first, MI-pix2pix, is a multi-input extension of the well known pix2pix approach [63] for image-to-image translation problems. The second, MI-GAN, was adapted from the approach introduced by Sharma et al. in 2019 [64]. These two models were also compared to a single-input pix2pix model to assess better the benefits of using a multi-input approach. We trained these models to generate missing modalities for brain MRIs using the BraTS2015 dataset. We designed a set of quantitative metrics to assess the different approaches' performance and performed a qualitative analysis. Our results show that generated images are relatively accurate and realistic compared to real images available in the dataset. In some cases, it might be difficult to distinguish between real and generated images. We also tested our approach using both a uni-modal – for the FLAIR modality – and a multi-modal SegAN-CAT model. Our results show that the generated images could be used to solve segmentation tasks by reaching a relatively good performance compared to using authentic images. However, our analysis on multi-input SegAN-CAT highlighted the importance of considering the target model for which the images are generated. A multi-input model can learn inherently more complex patterns than single-input models, including features that are not easily recognizable by visual inspection and may mislead the model. Due to the black-box nature of CNNs – and the complex

nature of adversarial networks – a direct investigation could be difficult to perform. Nonetheless, we envision that further research on this topic might help better understand the problem. To conclude, our findings suggest that (i) multi-input models generally perform better than single-input ones with a few exceptions and (ii) the MI-pix2pix model allows us to achieve, based on our quantitative metrics, better results than MI-GAN.

CHAPTER 7

Conclusions

In our study, we addressed the problem of collaborative machine learning for healthcare imaging. Collaborative machine learning is a set of techniques and paradigms that enable sharing of knowledge between different parties – e.g., different hospitals or research institutes – that hold heterogeneous datasets or datasets too small to apply machine learning techniques successfully. First, we discussed the major issues in applying machine learning and AI to healthcare. The first issue is related to data availability and standardization. While data collection in hospitals is performed daily, building shared datasets is a complex and costly task, as it requires shared data collection policies and procedures. Only a few public datasets are large enough to successfully apply advanced machine learning techniques like Deep Learning in medical imaging. Publishing such large datasets also requires a great effort in terms of human resources and legal processes. Data Imbalance and Heterogeneity represent another set of issues: data coming from different institutions are inevitably processed in different ways due to different equipment, acquiring processes standards, and other practical and organizational constraints. Moreover, datasets collected from different sources are inherently imbalanced in terms of samples due to population distribution and the institution's size. Another issue is privacy, as health

records are classified as sensitive data and require special treatment. However, legislation varies from country to country, and hospitals may also have internal, more stringent policies. This issue is also related to the security issue, intended as the need for mechanisms for keeping data used for learning undisclosed to unauthorized parties.

We introduced standard machine learning models, such as neural networks and decision trees. Later, we defined some of the most popular machine learning tasks and their application in healthcare imaging. In particular, the classification task of automating the diagnosis of medical images is particularly relevant to the field. It helps physicists perform diagnoses with less effort and with more confidence. Another prevalent machine learning task in medical imaging is related to segmentation and localization, which allows automating the process of identifying which pixels or voxels in an image belong to a particular region of interest – e.g., a lesion. A relatively new application of machine learning in medical imaging is image translation. This technique allows generating an image, starting from another one used as an input. Thus, it is possible to generate either a different kind of imaging – e.g., CT to PET images-, or a different contrast modality – e.g., T_1 MRI to T_2 MRI. To this extent, we considered an essential task of practical relevance: commonly acquired datasets of multi-modality images – e.g., MRI – are often incomplete, meaning that not every modality is available for each patient, and this constitutes an issue if the available machine learning model requires all the modalities in input. Image translation can be used to this extent to solve the problem efficiently.

Envisioning a collaborative machine learning approach, we considered a *distributed learning* approach to address the issues mentioned above. To this extent, we analyzed the state of the art of distributed learning applied to medical imaging and healthcare and proposed a framework that defines different dimensions of analysis. The three major paradigms we found in the literature are *Ensemble Learning*, *Split Learning*, *Federated Learning*. In particular, we classified the different paradigms according to the (i) number of models in output, (ii) the ownership of the weights – e.g., Local, Centralized or both–, and (iii) the nature of data exchanged by the nodes. Then, we analyzed each contribution more in-depth, considering the specific applicative domain, the machine learning model used, the distributed approach, and the sources of data imbalance and heterogeneity. From our survey, the most used approaches are Federated Learning and Incremental Learning, which can be seen as a particular case of FL. The contributions that compare different distributed approaches mainly compare these two approaches, other than with the baselines given by either *centralized* – e.g.,

centralized dataset –or *local* – i.e., learning without information sharing– solutions. Being CNN a model particularly suitable for images, we envision that a comparison including Split Learning could be beneficial.

Ensemble Learning. To study the problem of collaborative learning from the perspective introduced above, we consider a scenario in which different classification or segmentation models are available. For this reason, we considered an *ensemble learning* paradigm to combine multiple models in a single one. In particular, we considered two different settings: in the first scenario, we address the problem of heterogeneous data. Different models are trained on different data distributions or different imaging modalities. In the second scenario, we consider an ensemble learning approach when the heterogeneity resides in the model architecture rather than data.

We first introduced two sets of ensembling techniques for segmentation and classification, one based on *simple average*, *maximum proposal*, *majority voting* and the other based on the *entropy-weighted average* of the predictions, calculated on different portions of the output. Then, for the first scenario we first tested our approach on a dataset of *simulated proposals*, to test different *agent* configurations in the case of one *expert* agent *vs.* N agents that are biased on either one of the two classes. We performed our analysis considering the number of agents in the system, the performances of the expert agent, and the noise that could affect the predictions. Then, we introduced real segmentation models based on CNN, testing the approach both on non-medical images – to overcome the lack of segmented data – and on the BraTS dataset. Lastly, we performed an analysis considering two additional dimensions for each segmentation case: the number of *confident* agents and the number of *correct predictions*. Our results show that, depending on the relative performances of the agents, our methods based on *entropy-weighted average* are a successful approach in many cases. However, we also show that the relative performances of each method are highly dependent on several factors –e.g., the confidence of the models or their behavior on irregular patterns.–, which may not always be fully captured by a simulated analysis. Nonetheless, we provided a preliminary overview of the possible application of each ensembling method on a segmentation task according to different conditions, although limited by the scarcity of relevant datasets and pre-trained models.

Considering the second scenario, we applied three different ensembling techniques on Automated Chest X-Ray Diagnosis. We trained seven different CNN architectures on the CheXpert dataset to this extent. We used a simple average, an entropy-weighted average, and a stacking approach

to obtain a single classifier from the seven models. Our results show that the entropy-weighted average can outperform the best CNN architecture for each label in 4 cases out of 5. Moreover, since no CNN was able to outperform the others on every label, ensembling learning methods proved to be successful in combining the strengths of each model.

Distributed Learning. As highlighted by our survey on state-of-the-art, a popular approach in distributed learning applied to healthcare is the Federated Learning paradigm. In addition, Split Learning is another distributed paradigm that has been designed specifically for healthcare settings. One of the differences between ensembling and distributed learning methods is that for the latter, the exchange of information happens during the training phase of the model. For this reason, when a task is designed to be solved collaboratively from the beginning, FL and SL constitute a viable design solution. However, to our knowledge, no direct comparisons of the two methods have been performed on medical imaging classification with a focus on heterogeneity. To this extent, we proposed a comparison between the two methods, taking into account: (i) the heterogeneity of the data, in terms of label distribution, (ii) the frequency of data sharing for the update of the models, (iii) the choice of the architecture, in the case of Split Learning. Our results show that both methods can outperform the models trained on local datasets, motivating the choice of a distributed learning solution. As confirmed by previous findings, Federated Learning can reach performances comparable to those of a centralized solution. Split Learning, instead, allows for more design versatility while providing good performances overall. When introducing data heterogeneity in the form of unbalanced datasets, performances of SL are slightly more affected than those of FL. We envision, however, that this issue can be compensated by a more careful design of the local models, as one of the advantages of Split Learning is that it allows the different client models to be customized according to specific needs. Lastly, as expected, introducing a U-Shaped architecture in Split Learning allows enabling privacy on the exchanged labels at the cost of a minor loss in performances. Distributed Learning approaches are an active field of research; for this reason, they may require more effort to be implemented compared to other distributed learning solutions such as ensemble learning and transfer learning. However, our results confirm them as valuable tools in the design of collaborative learning systems.

Transfer Learning. The third paradigm we analyzed is Transfer Learning, a well-known approach that allows training new machine learning models using other models trained on different data. However, it also constitutes an important tool to be used in a collaborative setting, especially

when privacy regulations allow only to exchange model parameters. In addition, transfer learning is versatile enough to address the problem of diverse nature, such as missing modalities.

The first use of Transfer Learning we investigated is related to exploiting the feature extraction pipelines of our models trained on CheXpert, to build a new set of classifiers based on trees. To do this, we used a technique derived from text-based machine learning models, called *image embedding*. Embeddings are a compact representation of input data extracted from a CNN. Once we extracted the embeddings, we proceeded in training two sets of classifiers based on Random Forests and XGBoost. Once more, no single model can outperform the others on every label, so we resorted again to ensembling to produce a global model that accounted for all our CheXpert models. Our results show that embeddings can capture features from the data that are relevant to the task. Moreover, our models based on trees can outperform the previous classifiers based on neural networks. This indicates that while CNNs are a promising approach to perform feature extraction, neural network classifiers may require more data than other classifiers to provide good performances.

Our previous results addressed the problem of heterogeneity in data and the model architecture. Lastly, we exploit transfer learning to address the target labels' heterogeneity by applying a domain adaptation approach. To this extent, introduced a new local dataset of Chest X-Rays, called HUM-CXR, that includes different labels from CheXpert. Then, we exploited our models trained on CheXpert to investigate how transfer learning can train a classifier on the HUM-CXR dataset. We investigate different strategies, including the classical fine-tuning of the pre-trained CNNs and learning a new set of tree classifiers on the embeddings generated by the CheXpert CNNs. Our results show that the approach based on embeddings has the advantage of being both the one with the highest performances and the less resource-intensive. This result confirms the potential of our approach in transferring relevant features to datasets with a different distribution –e.g., data sourced from a different hospital, with different population and label definitions. In addition, we performed an analysis based on Explainable AI (XAI), using a GradCAM approach. This technique allowed investigating which areas in the input images the network focuses on to predict the various labels. Our analysis on 200 CXRs showed that the network correctly focuses on the lung sector when producing the classification. However, for the *device* class, we discovered that when the patient has a pacemaker, the network often focuses on the leads that go inside the heart instead of focusing on the device box. We envision that this may be due to biases present

in the CheXpert dataset; however, we could not investigate the issue further since the CheXpert dataset does not report the percentage of each device type.

In the last set of experiments, we instead focused on the problem of *Missing Modality Generation* in MRI segmentation datasets. While in public datasets commonly used for research, the different MR modalities are present for every patient, this does not always happen in clinical practice. When considering segmentation models, it may be beneficial to take advantage of models that take multiple modalities as input, as they can exploit the relations between the different modalities and produce better results. However, this could not be done when a modality is missing from the data.

To address this issue, we first proposed an Adversarial Network model for segmentation, called SegAN-CAT, based on a model proposed in the literature. In particular, the goals of this experiment was to (i) compare our SegAN-CAT model with the SegAN model proposed by Xue et al. [185], (ii) assess the performances of the uni-modal vs. multi-modal versions of SegAN-CAT, and (iii) study the optimal strategy to perform transfer learning across the different modalities. Our results show how our approach allows improving *SegAN* performances on both BraTS 2015 and BraTS 2019 datasets. Moreover, none of the uni-modal networks could reach the same performances of the model trained using all the four input modalities. However, the model trained on the FLAIR modality was the one closest to the performances of our multi-modality model. For this reason, we investigated how to exploit transfer learning to train a model that performs better on other MRI modalities. To this extent, we tested different fine-tuning approaches. Despite the difficulties in fine-tuning an Adversarial Network [225, 227], our results show that it is possible to transfer a model trained on FLAIR to improve the performances of a model trained on other modalities compared to the same model trained from scratch.

One of the possible strategies for enabling multi-input models for segmentation in an incomplete dataset is to exploit image translation to generate missing modalities from the available ones. To this extent, we investigated two different approaches based on GANs. In particular, our two approaches use two different architectures to generate one target modality from the three available ones. We designed a set of quantitative metrics to assess the performances of our approaches and performed both a quantitative and qualitative analysis. Our results show that our image-translation approaches can generate images that are often visually indistinguishable from their real counterpart. Moreover, results suggest that a translation approach based on multiple input modalities is beneficial in many cases,

depending on the considered modalities. As a proof-of-concept, we tested our approach using our generated images as input to both a uni-modal and a multi-modal SegAN-CAT to perform segmentation. The resulting performances of the segmentation model when using generated images confirm that it is possible to perform segmentation using our generated images, confirming that the image translation process retains features relevant for segmentation.

7.1 Future Works

Our experiments investigated the advantages of collaborative learning for Medical Imaging. However, complex dynamics can arise from the approaches presented in this work. In Chapter 2, we proposed an analysis focused on providing a coherent taxonomy of the various distributed learning approaches. Our results show that only a few works tried to apply these methods to medical imaging, while a more significant amount applied distributed learning to medical records, such as EHR. We envision that a general framework for multi-centric studies on machine learning could help better understand what solutions and approaches are still to explore.

While providing fewer privacy guarantees by default, ensemble learning theory could provide valuable insights when combined with other methods, such as aggregating intermediate outputs in Split Learning or gradients for Federated Learning. Our experiments on ensembling methods applied to segmentation – although preliminary – showed that the dynamics are affected by different design variables. This variability suggests that ensembling methods for segmentation are more complex than the approaches used for classification. They deserve more thorough analysis and more careful modeling of each case study. On the other hand, our experiments that exploit ensemble techniques to aggregate different machine learning models for classification showed more consistent and coherent results, highlighting the validity of including a measure of confidence in the aggregation methods. However, this approach could be applied more accurately if a measure of confidence is available directly from the machine learning model itself. For this reason, we envision that this kind of method could be applied using a more comprehensive range of different machine learning models.

At the same time, we believe that future works are necessary to investigate better how distributed learning methods can perform with a smaller amount of data – as commonly happens in healthcare applications –. One major issue we faced in implementing our Split and Federated learning comparison is the lack of related modules in popular machine learning

frameworks, as Tensorflow [243]. While for Federated Learning, the *TensorFlow Federated* module is integrated into the code repository, in our experiments, we experienced some issues related to code stability between the different releases. Concerning Split Learning, to the best of our knowledge, a similar package still does not exist; for this reason, we had to develop a custom implementation that currently does not support training using different physical machines. Moreover, due to computational cost associated with Federated and Split Learning when deployed to a single host, we had to use a less resource-intensive architecture, affecting the global performances of the model. We envision that further optimization and research on the topic could allow a more straightforward deployment of such methods, allowing the use of more complex and better-performing machine learning architectures.

The use of AI and radiomics is an active field of research; however, their maturity is not yet enough to be applied in standard practice [244]. Concerning feature extraction, we also envision the possibility of combining radiomics features with CNN-extracted features [93], by either providing radiomics features as input to a neural network or using CNNs to produce radiomic features [245]. Our experiments focused on a CNN approach by applying embedding and ensembling. Our results showed that it is possible to train different classifiers by exploiting the feature extraction step provided by Convolutional Neural Network. However, the features included in the embeddings are only relevant for solving the task on which the CNN has been previously trained. In other words, the embeddings extracted for a network trained to solve a source task might not be adequate to train models to solve a very different task. In one ongoing work, we are approaching this issue by investigating the possibility to exploit Variational Autoencoders (VAE) [73] instead of CNN to perform the feature extraction task. Variational Autoencoders are a particular kind of neural network trained to learn a compact representation of the features that describe input data. In this case, the VAE could produce a general-purpose feature extraction method that can be used to perform transfer learning to different tasks – e.g., prognosis prediction, target localization, etc.– in a similar fashion to our experiments with the HUM-XRAY dataset.

Another issue worth more investigation is related to dataset bias. Due to the difficulty of collecting datasets with different distributions – e.g., different demographic and geographic characteristics–, studies based on Deep Learning often use training and testing splits of the same dataset. One example is CheXpert [246], [247], in which data are collected from Stanford Hospital. This practice could limit their transition to clinical application

[248]. Weber et al. [249] discussed the importance of evaluating the performance of a Deep Learning model on applications they were not explicitly trained for to characterize their generalization capabilities. In the summary of Chapter 6, we also discussed that a further effort in defining the ground truth might help discriminate some classes useful in medical practice - such as patient devices.

In our experiments on segmentation, we investigated a transfer learning approach based on Adversarial Network. In particular, applying transfer learning to train adversarial networks is a challenging and yet rather unexplored research area that deserves additional investigation. Moreover, our analysis had been limited by the availability of pre-trained Adversarial Networks related to the task. We envision that the availability of pre-trained network models that include both the generator and discriminator weights might be a successful approach in enabling transfer learning approaches for adversarial networks. Another important topic relevant to adversarial networks is the issue of introducing a better qualitative assessment of the generated samples. In our experiments on image translation, we proposed an approach that exploits our segmentation model to assess modality generation performances. However, our results seem to depend on the network used for evaluation. We envision that further research on the topic and the design of more specific assessment metrics could further improve the field of image generation for medical imaging.

In our study, we provided insights on how a collaborative learning approach could be exploited to overcome the issues of data availability and heterogeneity. Our research has been limited primarily by the issue of data availability. However, another important aspect is the availability of pre-trained models. While it is an established practice to exploit pre-trained models for general imaging, e.g., using the Keras-Application module [175] or Tensorflow Hub [250], a similar approach is more difficult when considering medical imaging tasks. While interesting projects exist, they have often discontinued support – e.g., NiftyNet [251] – or are currently in development – e.g., MONAI [252]. Thus, we envision that more effort on this topic could significantly benefit the related research.

7.1.1 Equipment

Experiments on segmentation – both in Chapter 4 and 6–, have been performed using the Tensorflow Docker environment [243, 253, 254] [1] on a

[1] We used different Tensorflow versions across the experiments. In particular, the 2.0a version for the SegAN-CAT trained on BraTS 2015 and 2.0b for BraTS 2019, while we used Tensorflow 2.1 for the other experiments on Colab. Particular care has been put in ensuring that the framework development did not affect our results; in par-

server equipped with a 12GB Nvidia Titan V, Intel Xeon E5-2609 CPU and 64GB of RAM. All the other experiments have been performed using Google Colaboratory on an Nvidia Tesla P100-PCI-E-16GB GPU with 26 GB of RAM and an Intel(R) Xeon(R) CPU @ 2.30GHz.

ticular, mixing models trained on 2.0a and 2.0b provided wrong results due to a software bug in a normalization layer.

Bibliography

[1] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[2] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S. Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona J. Gilbert, Mark Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher J. Kelly, Dominic King, Joseph R. Ledsam, David Melnick, Hormuz Mostofi, Lily Peng, Joshua Jay Reicher, Bernardino Romera-Paredes, Richard Sidebottom, Mustafa Suleyman, Daniel Tse, Kenneth C. Young, Jeffrey De Fauw, and Shravya Shetty. International evaluation of an ai system for breast cancer screening. *Nature*, 577, 1 2020. ISSN 0028-0836. doi: 10.1038/s41586-019-1799-6.

[3] Kunio Doi. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. *Computerized medical imaging and graphics*, 31(4-5):198–211, 2007.

[4] Martina Sollini, Francesco Bartoli, Andrea Marciano, Roberta Zanca, Riemer H. J. A. Slart, and Paola A. Erba. Artificial intelligence and hybrid imaging: the best match for personalized medicine in oncology. *European Journal of Hybrid Imaging*, 4, 12 2020. ISSN 2510-3636. doi: 10.1186/s41824-020-00094-8.

[5] Christopher J. Kelly, Alan Karthikesalingam, Mustafa Suleyman, Greg Corrado, and Dominic King. Key challenges for delivering clinical impact with artificial intelligence. *BMC Medicine*, 17, 12 2019. ISSN 1741-7015. doi: 10.1186/s12916-019-1426-2.

[6] Gerold Porenta. Is there value for artificial intelligence applications in molecular imaging and nuclear medicine? *Journal of Nuclear Medicine*, 60, 10 2019. ISSN 0161-5505. doi: 10.2967/jnumed.119.227702.

[7] Julia Powles and Hal Hodson. Google deepmind and healthcare in an age of algorithms. *Health and technology*, 7(4):351–367, 2017.

[8] Samuel J Wang, Blackford Middleton, Lisa A Prosser, Christiana G Bardon, Cynthia D Spurr, Patricia J Carchidi, Anne F Kittler, Robert C Goldszer, David G Fairchild, Andrew J Sussman,

et al. A cost-benefit analysis of electronic medical records in primary care. *The American journal of medicine*, 114(5):397–403, 2003.

[9] Michael A Tutty, Lindsey E Carlasare, Stacy Lloyd, and Christine A Sinsky. The complex case of EHRs: examining the factors impacting the EHR user experience. *Journal of the American Medical Informatics Association*, 26(7):673–677, 04 2019. ISSN 1527-974X. doi: 10.1093/jamia/ocz021. URL `https://doi.org/10.1093/jamia/ocz021`.

[10] Peter Mildenberger, Marco Eichelberg, and Eric Martin. Introduction to the dicom standard. *European radiology*, 12(4):920–927, 2002.

[11] John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric Karl Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS medicine*, 15(11):e1002683, 2018.

[12] Anne Marie McCarthy, Brad M Keller, Lauren M Pantalone, Meng-Kang Hsieh, Marie Synnestvedt, Emily F Conant, Katrina Armstrong, and Despina Kontos. Racial differences in quantitative measures of area and volumetric breast density. *Journal of the National Cancer Institute*, 108(10):djw104, 2016.

[13] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical image analysis*, 58:101552, 2019.

[14] W Nicholson Price and I Glenn Cohen. Privacy in the age of medical big data. *Nature medicine*, 25(1):37–43, 2019.

[15] Mark J Taylor and James Wilson. Reasonable expectations of privacy and disclosure of health data. *Medical law review*, 27(3):432–460, 2019.

[16] Bradley Malin and Latanya Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of biomedical informatics*, 37(3):179–192, 2004.

[17] Samuel G Finlayson, Hyung Won Chung, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. *arXiv preprint arXiv:1804.05296*, 2018.

[18] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[19] Margarita Kirienko, Martina Sollini, Gaia Ninatti, Daniele Loiacono, Edoardo Giacomello, Noemi Gozzi, Francesco Amigoni, Luca Mainardi, Pier Luca Lanzi, and Arturo Chiti. Distributed learning: a reliable privacy-preserving strategy to change multicenter collaborations using ai. *European Journal of Nuclear Medicine and Molecular Imaging*, pages 1–14, 2021.

[20] Edoardo Giacomello, Daniele Loiacono, and Luca Mainardi. Brain mri tumor segmentation with adversarial networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[21] Emanuel Alogna, Edoardo Giacomello, and Daniele Loiacono. Brain magnetic resonance imaging generation using generative adversarial networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2528–2535. IEEE, 2020.

[22] Edoardo Giacomello, Pier Luca Lanzi, Daniele Loiacono, and Luca Nassano. Image embedding and model ensembling for automated chest x-ray interpretation. *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[23] Edoardo Giacomello, Michele Cataldo, Daniele Loiacono, and Pier Luca Lanzi. Distributed learning approaches for automated chest x-ray diagnosis. *arXiv preprint arXiv:2110.01474*, 2021.

[24] James H Cole, Rudra PK Poudel, Dimosthenis Tsagkrasoulis, Matthan WA Caan, Claire Steves, Tim D Spector, and Giovanni Montana. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163:115–124, 2017.

[25] Uran Ferizi, Harrison Besser, Pirro Hysi, Joseph Jacobs, Chamith S Rajapakse, Cheng Chen, Punam K Saha, Stephen Honig, and Gregory Chang. Artificial intelligence applied to osteoporosis: a performance comparison of machine learning algorithms in predicting fragility fractures from mri data. *Journal of Magnetic Resonance Imaging*, 49(4):1029–1038, 2019.

[26] Germán González, George R Washko, and Raúl San José Estépar. Deep learning for biomarker regression: application to osteoporosis and emphysema on chest ct scans. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105741H. International Society for Optics and Photonics, 2018.

[27] Thao P Ho-Le, Jacqueline R Center, John A Eisman, Tuan V Nguyen, and Hung T Nguyen. Prediction of hip fracture in post-menopausal women using artificial neural network approach. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4207–4210. IEEE, 2017.

[28] Christian Kruse, P Eiken, and P Vestergaard. Clinical fracture risk evaluated by hierarchical agglomerative clustering. *Osteoporosis International*, 28(3):819–832, 2017.

[29] Uran Ferizi, Stephen Honig, and Gregory Chang. Artificial intelligence, osteoporosis and fragility fractures. *Current opinion in rheumatology*, 31(4):368, 2019.

[30] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn L. Ball, Katie S. Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *CoRR*, abs/1901.07031, 2019. URL `http://arxiv.org/abs/1901.07031`.

[31] Spyridon Bakas, Mauricio Reyes, András Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Takeshi Shinohara, Christoph Berger, Sung Min Ha, Martin Rozycki, Marcel Prastawa, Esther Alberts, Jana Lipková, John B. Freymann, Justin S. Kirby, Michel Bilello, Hassan M. Fathallah-Shaykh, Roland Wiest, Jan Kirschke, Benedikt Wiestler, Rivka R. Colen, Aikaterini Kotrotsou, Pamela LaMontagne, Daniel S. Marcus, Mikhail Milchenko, Arash Nazeri, Marc-André Weber, Abhishek Mahajan, Ujjwal Baid, Dongjin Kwon, Manu Agarwal, Mahbubul Alam, Alberto Albiol, Antonio Albiol, Alex Varghese, Tran Anh Tuan, Tal Arbel, Aaron Avery, Pranjal B., Subhashis Banerjee, Thomas Batchelder, Kayhan N. Batmanghelich, Enzo Battistella, Martin Bendszus, Eze Benson, José Bernal, George Biros, Mariano Cabezas, Siddhartha Chandra, Yi-Ju Chang, and et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BraTS challenge. *CoRR*, abs/1811.02629, 2018. URL `http://arxiv.org/abs/1811.02629`.

[32] Ellery Wulczyn, David F Steiner, Zhaoyang Xu, Apaar Sadhwani, Hongwu Wang, Isabelle Flament-Auvigne, Craig H Mermel, Po-Hsuan Cameron Chen, Yun Liu, and Martin C Stumpe. Deep learning-based survival prediction for multiple cancer types using histopathology images. *PLoS One*, 15(6):e0233678, 2020.

[33] Po-Yu Kao, Thuyen Ngo, Angela Zhang, Jefferson W Chen, and BS Manjunath. Brain tumor segmentation and tractographic feature extraction from structural mr images for overall survival prediction. In *International MICCAI Brainlesion Workshop*, pages 128–141. Springer, 2018.

[34] Christian Kruse, Pia Eiken, and Peter Vestergaard. Machine learning principles can improve hip fracture prediction. *Calcified tissue international*, 100(4):348–360, 2017.

[35] Alexander de Brebisson and Giovanni Montana. Deep neural networks for anatomical brain segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[36] Mahsa Shakeri, Stavros Tsogkas, Enzo Ferrante, Sarah Lippe, Samuel Kadoury, Nikos Paragios, and Iasonas Kokkinos. Sub-cortical brain structure segmentation using f-cnn's. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 269–272. IEEE, 2016.

[37] Wenlu Zhang, Rongjian Li, Houtao Deng, Li Wang, Weili Lin, Shuiwang Ji, and Dinggang Shen. Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, 108:214–224, 2015.

[38] Ariel Birenbaum and Hayit Greenspan. Longitudinal multiple sclerosis lesion segmentation using multi-view convolutional neural networks. In *Deep Learning and Data Labeling for Medical Applications*, pages 58–67. Springer, 2016.

[39] Amitava Halder, Debangshu Dey, and Anup K Sadhu. Lung nodule detection from feature engineering to deep learning in thoracic ct images: a comprehensive review. *Journal of digital imaging*, 33(3):655–677, 2020.

[40] Shikha Agrawal and Jitendra Agrawal. Neural network techniques for cancer prediction: A survey. *Procedia Computer Science*, 60:769–774, 2015.

[41] Ian Middleton and Robert I Damper. Segmentation of magnetic resonance images using a combination of neural networks and active contour models. *Medical engineering & physics*, 26(1):71–86, 2004.

[42] Peijun Hu, Fa Wu, Jialin Peng, Yuanyuan Bao, Feng Chen, and Dexing Kong. Automatic abdominal multi-organ segmentation using deep convolutional neural network and time-implicit level sets. *International journal of computer assisted radiology and surgery*, 12(3):399–411, 2017.

[43] Nuo Tong, Shuiping Gou, Shuyuan Yang, Dan Ruan, and Ke Sheng. Fully automatic multi-organ segmentation for head and neck cancer radiotherapy using shape representation model constrained fully convolutional neural networks. *Medical physics*, 45(10):4558–4567, 2018.

[44] Yang Lei, Yabo Fu, Tonghe Wang, Richard LJ Qiu, Walter J Curran, Tian Liu, and Xiaofeng Yang. Deep learning in multi-organ segmentation. *arXiv preprint arXiv:2001.10619*, 2020.

[45] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL `http://dx.doi.org/10.1007/s11263-019-01228-7`.

[46] Maciej A Mazurowski, Mateusz Buda, Ashirbani Saha, and Mustafa R Bashir. Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on mri. *Journal of magnetic resonance imaging*, 49(4):939–954, 2019.

[47] David Rey, Gérard Subsol, Hervé Delingette, and Nicholas Ayache. Automatic detection and segmentation of evolving processes in 3d medical images: Application to multiple sclerosis. *Medical image analysis*, 6(2):163–179, 2002.

[48] Holger R Roth, Le Lu, Ari Seff, Kevin M Cherry, Joanne Hoffman, Shijun Wang, Jiamin Liu, Evrim Turkbey, and Ronald M Summers. A new 2.5 d representation for lymph node detection using random sets of deep convolutional neural network observations. In *International conference on medical image computing and computer-assisted intervention*, pages 520–527. Springer, 2014.

[49] Qi Dou, Hao Chen, Lequan Yu, Lin Shi, Defeng Wang, Vincent CT Mok, and Pheng Ann Heng. Automatic cerebral microbleeds detection from mr images via independent subspace analysis based hierarchical features. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 7933–7936. IEEE, 2015.

[50] Qi Dou, Hao Chen, Lequan Yu, Lei Zhao, Jing Qin, Defeng Wang, Vincent CT Mok, Lin Shi, and Pheng-Ann Heng. Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks. *IEEE transactions on medical imaging*, 35(5):1182–1195, 2016.

[51] Zhantao Cao, Lixin Duan, Guowu Yang, Ting Yue, Qin Chen, Huazhu Fu, and Yanwu Xu. Breast tumor detection in ultrasound images using deep learning. In *International Workshop on Patch-based Techniques in Medical Imaging*, pages 121–128. Springer, 2017.

[52] Richard Platania, Shayan Shams, Seungwon Yang, Jian Zhang, Kisung Lee, and Seung-Jong Park. Automated breast cancer diagnosis using deep learning and region of interest detection (bc-droid). In *Proceedings of the 8th ACM international conference on bioinformatics, computational biology, and health informatics*, pages 536–543, 2017.

[53] Ravi K Samala, Heang-Ping Chan, Lubomir Hadjiiski, Mark A Helvie, Jun Wei, and Kenny Cha. Mass detection in digital breast tomosynthesis: Deep convolutional neural network with transfer learning from mammography. *Medical physics*, 43(12):6654–6666, 2016.

[54] Holger R Roth, Le Lu, Jiamin Liu, Jianhua Yao, Ari Seff, Kevin Cherry, Lauren Kim, and Ronald M Summers. Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE transactions on medical imaging*, 35(5):1170–1181, 2015.

[55] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[56] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[57] Ning Li, Haopeng Liu, Bin Qiu, Wei Guo, Shijun Zhao, Kungang Li, and Jie He. Detection and attention: diagnosing pulmonary lung cancer from ct by imitating physicians. *arXiv preprint arXiv:1712.05114*, 2017.

[58] Ruhan Sa, William Owens, Raymond Wiegand, Mark Studin, Donald Capoferri, Kenneth Barooha, Alexander Greaux, Robert Rattray, Adam Hutton, John Cintineo, et al. Intervertebral disc detection in x-ray images using faster r-cnn. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 564–567. IEEE, 2017.

[59] Holger R Roth, Yinong Wang, Jianhua Yao, Le Lu, Joseph E Burns, and Ronald M Summers. Deep convolutional networks for automated detection of posterior-element fractures on spine ct. In *Medical imaging 2016: computer-aided diagnosis*, volume 9785, page 97850P. International Society for Optics and Photonics, 2016.

[60] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.

[61] Jiamin Liu, David Wang, Le Lu, Zhuoshi Wei, Lauren Kim, Evrim B Turkbey, Berkman Sahiner, Nicholas A Petrick, and Ronald M Summers. Detection and diagnosis of colitis on computed tomography using deep convolutional neural networks. *Medical physics*, 44(9): 4630–4642, 2017.

[62] Gabriel Efrain Humpire-Mamani, Arnaud Arindra Adiyoso Setio, Bram van Ginneken, and Colin Jacobs. Efficient organ localization using multi-label convolutional neural networks in thorax-abdomen ct scans. *Physics in Medicine & Biology*, 63(8):085003, 2018.

[63] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016. URL `http://arxiv.org/abs/1611.07004`.

[64] Anmol Sharma and Ghassan Hamarneh. Missing MRI Pulse Sequence Synthesis using Multi-Modal Generative Adversarial Network. *arXiv e-prints*, art. arXiv:1904.12200, Apr 2019.

[65] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL `http://arxiv.org/abs/1703.10593`.

[66] Salman Ul Hassan Dar, Mahmut Yurt, Levent Karacan, Aykut Erdem, Erkut Erdem, and Tolga Çukur. Image synthesis in multi-contrast MRI with conditional generative adversarial networks. *CoRR*, abs/1802.01221, 2018. URL `http://arxiv.org/abs/1802.01221`.

[67] B. Yu, L. Zhou, L. Wang, J. Fripp, and P. Bourgeat. 3d cgan based cross-modality mr image synthesis for brain tumor segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 626–630, April 2018. doi: 10.1109/ISBI.2018.8363653.

[68] Mauricio Orbes-Arteaga, M. Jorge Cardoso, Lauge Sørensen, Marc Modat, Sébastien Ourselin, Mads Nielsen, and Akshay Pai. Simultaneous synthesis of FLAIR and segmentation of white matter hypointensities from T1 mris. *CoRR*, abs/1808.06519, 2018. URL `http://arxiv.org/abs/1808.06519`.

[69] C. Ge, I. Y. Gu, A. Store Jakola, and J. Yang. Cross-modality augmentation of brain mr images using a novel pairwise generative adversarial network for enhanced glioma classification. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 559–563, Sep. 2019. doi: 10.1109/ICIP.2019.8803808.

[70] Mohammad Havaei, Nicolas Guizard, Nicolas Chapados, and Yoshua Bengio. Hemis: Hetero-modal image segmentation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 469–477, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46723-8.

[71] Thomas Varsavsky, Zach Eaton-Rosen, Carole H. Sudre, Parashkev Nachev, and M. Jorge Cardoso. PIMMS: permutation invariant multi-modal segmentation. *CoRR*, abs/1807.06537, 2018. URL `http://arxiv.org/abs/1807.06537`.

[72] Reuben Dorent, Samuel Joutard, Marc Modat, Sébastien Ourselin, and Tom Vercauteren. Hetero-Modal Variational Encoder-Decoder for Joint Modality Completion and Segmentation. *arXiv e-prints*, art. arXiv:1907.11150, Jul 2019.

[73] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, art. arXiv:1312.6114, Dec 2013.

[74] Food, Drug Administration, et al. Proposed regulatory framework for modifications to artificial intelligence/machine learning (ai/ml)-based software as a medical device (samd). 2019.

[75] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3145–3153. JMLR.org, 2017.

[76] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4768–4777. Curran Associates Inc., 2017. ISBN 9781510860964.

[77] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3395–3403. Curran Associates Inc., 2016. ISBN 9781510838819.

[78] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2016. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.319.

[79] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 10 2017. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.74.

[80] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10, 7 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140.

[81] Alex J. DeGrave, Joseph D. Janizek, and Su-In Lee. Ai for radiographic covid-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 5, 05 2021. doi: 10.1038/s42256-021-00338-7.

[82] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[83] Markus Svensén and Christopher M Bishop. Pattern recognition and machine learning, 2007.

[84] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[85] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[86] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[87] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[89] Pim Moeskops, Jelmer M. Wolterink, Bas H. M. van der Velden, Kenneth G. A. Gilhuijs, Tim Leiner, Max A. Viergever, and Ivana Isgum. Deep learning for multi-task medical image segmentation in multiple modalities. *CoRR*, abs/1704.03379, 2017. URL `http://arxiv.org/abs/1704.03379`.

[90] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[91] Robert J Gillies, Paul E Kinahan, and Hedvig Hricak. Radiomics: images are more than pictures, they are data. *Radiology*, 278(2):563–577, 2016.

[92] Michele Avanzo, Lise Wei, Joseph Stancanello, Martin Vallieres, Arvind Rao, Olivier Morin, Sarah A Mattonen, and Issam El Naqa. Machine and deep learning methods for radiomics. *Medical physics*, 47(5):e185–e202, 2020.

[93] Vishwa S Parekh and Michael A Jacobs. Deep learning and radiomics in precision medicine. *Expert review of precision medicine and drug development*, 4(2):59–72, 2019.

[94] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[95] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. URL http://arxiv.org/abs/1511.06434.

[96] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. URL http://arxiv.org/abs/1411.1784.

[97] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58:101552, Dec 2019. ISSN 1361-8415. doi: 10.1016/j.media.2019.101552. URL http://dx.doi.org/10.1016/j.media.2019.101552.

[98] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[99] Ziyu Wan, Bo Zhang, Dongdong Chen, Pan Zhang, Dong Chen, Jing Liao, and Fang Wen. Bringing old photos back to life. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2747–2757, 2020.

[100] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!*, pages 1–1. 2019.

[101] Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26:2814–2822, 2013.

[102] Weili Nie and Ankit B Patel. Towards a better understanding and regularization of gan training dynamics. In *Uncertainty in Artificial Intelligence*, pages 281–291. PMLR, 2020.

[103] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys (CSUR)*, 53(2):1–33, 2020.

[104] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[105] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[106] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. doi: 10.1109/34.58871.

[107] Anders Krogh and Jesper Vedelsby. Validation, and active learning. *Advances in neural information processing systems 7*, 7:231, 1995.

[108] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33(1):1–39, 2010.

[109] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. URL https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[110] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[111] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[112] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[113] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, MarcAurelio Ranzato, Andrew Senior, Paul Tucker, et al. Large scale distributed deep networks. 2012.

[114] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[115] Jakub Konecný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016. URL `http://arxiv.org/abs/1610.02527`.

[116] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. Goodle AI Blog, 2017. `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`.

[117] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016. URL `http://arxiv.org/abs/1610.05492`.

[118] Ken Chang, Niranjan Balachandar, Carson Lam, Darvin Yi, James Brown, Andrew Beers, Bruce Rosen, Daniel L Rubin, and Jayashree Kalpathy-Cramer. Distributed deep learning networks among institutions for medical imaging. *Journal of the American Medical Informatics Association*, 25(8):945–954, 2018.

[119] Samuel W Remedios, Snehashis Roy, Camilo Bermudez, Mayur B Patel, John A Butman, Bennett A Landman, and Dzung L Pham. Distributed deep learning across multisite datasets for generalized ct hemorrhage segmentation. *Medical physics*, 47(1):89–98, 2020.

[120] Micah J. Sheller, G. Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. *CoRR*, abs/1810.04304, 2018. URL `http://arxiv.org/abs/1810.04304`.

[121] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[122] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[123] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. *CoRR*, abs/1610.05202, 2016. URL `http://arxiv.org/abs/1610.05202`.

[124] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[125] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[126] Anand D Sarwate and Kamalika Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE signal processing magazine*, 30(5):86–94, 2013.

[127] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. Differential privacy-enabled federated learning for sensitive health data. *CoRR*, abs/1910.02578, 2019. URL http://arxiv.org/abs/1910.02578.

[128] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. https://eprint.iacr.org/2015/1192.

[129] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[130] Craig Gentry et al. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

[131] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical Report MSR-TR-2016-3, February 2016. URL https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughp

[132] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189, 2017. URL http://arxiv.org/abs/1711.05189.

[133] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Computing Surveys (CSUR)*, 53(4): 1–35, 2020.

[134] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[135] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4, page 9. San Diego, CA, USA, 2004.

[136] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3), 2017.

[137] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS 17, pages 619–631, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134056. URL https://doi.org/10.1145/3133956.3134056.

[138] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE, 2013.

[139] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies*, 2017(4):345–364, 2017.

[140] Junghye Lee, Jimeng Sun, Fei Wang, Shuang Wang, Chi-Hyuck Jun, and Xiaoqian Jiang. Privacy-preserving patient similarity learning in a federated environment: development and analysis. *JMIR medical informatics*, 6(2):e20, 2018.

[141] Theodora S. Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch. Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59–67, 2018. ISSN 1386-5056. doi: https://doi.org/10.1016/j.ijmedinf.2018.01.007. URL https://www.sciencedirect.com/science/article/pii/S138650561830008X.

[142] Rui Duan, Mary Regina Boland, Zixuan Liu, Yue Liu, Howard H Chang, Hua Xu, Haitao Chu, Christopher H Schmid, Christopher B Forrest, John H Holmes, Martijn J Schuemie, Jesse A Berlin, Jason H Moore, and Yong Chen. Learning from electronic health records across multiple sites: A communication-efficient and privacy-preserving distributed algorithm. *Journal of the American Medical Informatics Association*, 27(3):376–385, 12 2019. ISSN 1527-974X. doi: 10.1093/jamia/ocz199. URL https://doi.org/10.1093/jamia/ocz199.

[143] Li Huang and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *CoRR*, abs/1903.09296, 2019. URL http://arxiv.org/abs/1903.09296.

[144] Arthur Jochems, Timo M Deist, Johan Van Soest, Michael Eble, Paul Bulens, Philippe Coucke, Wim Dries, Philippe Lambin, and Andre Dekker. Distributed learning: developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept. *Radiotherapy and Oncology*, 121(3):459–467, 2016.

[145] Timo M Deist, Arthur Jochems, Johan van Soest, Georgi Nalbantov, Cary Oberije, Seán Walsh, Michael Eble, Paul Bulens, Philippe Coucke, Wim Dries, et al. Infrastructure and distributed learning methodology for privacy-preserving multi-centric rapid learning health care: eurocat. *Clinical and translational radiation oncology*, 4:24–31, 2017.

[146] Arthur Jochems, Timo M Deist, Issam El Naqa, Marc Kessler, Chuck Mayo, Jackson Reeves, Shruti Jolly, Martha Matuszak, Randall Ten Haken, Johan van Soest, et al. Developing and validating a survival prediction model for nsclc patients through distributed learning across 3 countries. *International Journal of Radiation Oncology* Biology* Physics*, 99(2):344–352, 2017.

[147] Timo M Deist, Frank JWM Dankers, Priyanka Ojha, M Scott Marshall, Tomas Janssen, Corinne Faivre-Finn, Carlotta Masciocchi, Vincenzo Valentini, Jiazhou Wang, Jiayan Chen, et al. Distributed learning on 20 000+ lung cancer patients–the personal health train. *Radiotherapy and Oncology*, 144:189–200, 2020.

[148] Fida K Dankar, Nisha Madathil, Samar K Dankar, and Sabri Boughorbel. Privacy-preserving analysis of distributed biomedical data: designing efficient and secure multiparty computations using distributed statistical learning theory. *JMIR medical informatics*, 7(2):e12702, 2019.

[149] Santiago Silva, Boris Gutman, Eduardo Romero, Paul M Thompson, Andre Altmann, and Marco Lorenzi. Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data, 2019.

[150] Samuel Remedios, Snehashis Roy, Justin A. Blaber, Camilo Bermudez, Vishwesh Nath, Mayur B. Patel, John A. Butman, Bennett A. Landman, and Dzung L. Pham. Distributed deep learning for robust multi-site segmentation of CT imaging after traumatic brain injury. *CoRR*, abs/1903.04207, 2019. URL `http://arxiv.org/abs/1903.04207`.

[151] Maarten G. Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *CoRR*, abs/1912.12115, 2019. URL `http://arxiv.org/abs/1912.12115`.

[152] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):1–12, 2020.

[153] P K Spiegel. The first clinical x-ray made in america–100 years. *American Journal of Roentgenology*, 164(1):241–243, 1995. doi: 10.2214/ajr.164.1.7998549. URL `https://doi.org/10.2214/ajr.164.1.7998549`. PMID: 7998549.

[154] Paul C Lauterbur. Image formation by induced local interactions: examples employing nuclear magnetic resonance. *nature*, 242(5394):190–191, 1973.

[155] Paul Suetens. *Fundamentals of medical imaging*. Cambridge university press, 2017.

[156] Donald G Mitchell and Mark S Cohen. Mri principles. 2004.

[157] Tomoko Okuda, Yukunori Korogi, Yoshinori Shigematsu, Takeshi Sugahara, Toshinori Hirai, Ichiro Ikushima, Luxia Liang, and Mutsumasa Takahashi. Brain lesions: when should fluid-attenuated inversion-recovery sequences be used in mr evaluation? *Radiology*, 212(3):793–798, 1999.

[158] Karen A Tong, S Ashwal, A Obenaus, JP Nickerson, D Kido, and EM Haacke. Susceptibility-weighted mr imaging: a review of clinical applications in children. *American Journal of Neuroradiology*, 29(1):9–17, 2008.

[159] Feng Chen and Yi-Cheng Ni. Magnetic resonance diffusion-perfusion mismatch in acute ischemic stroke: An update. *World journal of radiology*, 4(3):63, 2012.

[160] Christoph Stippich. *Clinical functional MRI: presurgical functional neuroimaging*. Springer, 2015.

[161] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL `http://arxiv.org/abs/1405.0312`.

[162] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John,

E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput. The multi-modal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, Oct 2015. doi: 10.1109/TMI.2014.2377694.

[163] Stanford ML Group. Chexpert competition. `https://stanfordmlgroup.github.io/competitions/chexpert/`.

[164] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison, 2019.

[165] Hieu H. Pham, Tung T. Le, Dat Q. Tran, Dat T. Ngo, and Ha Q. Nguyen. Interpreting chest x-rays via cnns that exploit disease dependencies and uncertainty labels, 2019.

[166] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, 2017.

[167] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[168] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.369. URL `http://dx.doi.org/10.1109/CVPR.2017.369`.

[169] Pranav Rajpurkar, Jeremy Irvin, Robyn L Ball, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis P Langlotz, et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists. *PLoS medicine*, 15(11):e1002686, 2018.

[170] Pulkit Kumar, Monika Grewal, and Muktabh Mayank Srivastava. Boosted cascaded convnets for multilabel classification of thoracic diseases in chest radiographs. In Aurélio Campilho, Fakhri Karray, and Bart ter Haar Romeny, editors, *Image Analysis and Recognition*, pages 546–552, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93000-8.

[171] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti. Multi-objective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2020. doi: 10.1109/TEVC.2020.3024708.

[172] Wenwu Ye, Jin Yao, Hui Xue, and Yi Li. Weakly supervised lesion localization with probabilistic-cam pooling, 2020.

[173] Jonathan Rubin, Deepan Sanghavi, Claire Zhao, Kathy Lee, Ashequl Qadir, and Minnan Xu-Wilson. Large scale automated reading of frontal and lateral chest x-rays using dual convolutional neural networks, 2018.

[174] Alistair E. W. Johnson, Tom J. Pollard, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih ying Deng, Yifan Peng, Zhiyong Lu, Roger G. Mark, Seth J. Berkowitz, and Steven Horng. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs, 2019.

[175] Keras Team. Keras applications. URL `https://keras.io/api/applications/`. Accessed: 2021-04-29 10:33:30.

[176] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[177] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

[178] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[179] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

[180] Michał Futrega, Alexandre Milesi, Michal Marcinkiewicz, and Pablo Ribalta. Optimized u-net for brain tumor segmentation. *arXiv preprint arXiv:2110.03352*, 2021.

[181] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *CoRR*, abs/1611.08408, 2016. URL `http://arxiv.org/abs/1611.08408`.

[182] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[183] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *2009 IEEE 12th international conference on computer vision*, pages 1–8. IEEE, 2009.

[184] Pim Moeskops, Mitko Veta, Maxime W. Lafarge, Koen A. J. Eppenhof, and Josien P. W. Pluim. Adversarial training and dilated convolutions for brain MRI segmentation. *CoRR*, abs/1707.03195, 2017. URL `http://arxiv.org/abs/1707.03195`.

[185] Yuan Xue, Tao Xu, Han Zhang, L. Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l_1 loss for medical image segmentation. *CoRR*, abs/1706.01805, 2017. URL `http://arxiv.org/abs/1706.01805`.

[186] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31, 2017.

[187] Konstantinos Kamnitsas, Christian Ledig, Virginia FJ Newcombe, Joanna P Simpson, Andrew D Kane, David K Menon, Daniel Rueckert, and Ben Glocker. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical image analysis*, 36:61–78, 2017.

[188] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016. URL `https://arxiv.org/abs/1604.04382`.

[189] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Aug 1995. doi: 10.1109/ICDAR.1995.598994.

[190] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

[191] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

[192] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[193] Yutaka Sasaki et al. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5, 2007.

[194] Thorvald Sørensen, TA Sørensen, TJ Sørensen, T SORENSEN, T Sorensen, TA Sorensen, and T Biering-Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. 1948.

[195] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26 (3):297–302, 1945.

[196] Jayawant N. Mandrekar. Receiver operating characteristic curve in diagnostic test assessment. *Journal of Thoracic Oncology*, 5(9):1315 – 1316, 2010. ISSN 1556-0864. doi: https://doi.org/10.1097/JTO.0b013e3181ec173d. URL http://www.sciencedirect.com/science/article/pii/S1556086415306043.

[197] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[198] Harold Cramer. Mathematical methods of statistics, princeton univ. *Press, Princeton, NJ*, 1946.

[199] Psnr. Mathworks. URL https://it.mathworks.com/help/vision/ref/psnr.html.

[200] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004. doi: 10.1109/TIP.2003.819861. URL https://ieeexplore.ieee.org/document/1284395.

[201] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[202] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[203] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2013.

[204] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *CoRR*, abs/1812.00564, 2018. URL http://arxiv.org/abs/1812.00564.

[205] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. https://arxiv.org/abs/1602.05629, 2017.

[206] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[207] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

[208] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

[209] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[210] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[211] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018.

[212] Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 7 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.9.

[213] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 6 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206848.

[214] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 10 2018. URL http://arxiv.org/abs/1810.04805.

[215] Chuong B Do and Andrew Y Ng. Transfer learning for text classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pages 299–306. MIT Press, 2005.

[216] Viswa Mani Kiran Peddinti and Prakriti Chintalapoodi. Domain adaptation in sentiment analysis of twitter. In *Proceedings of the 5th AAAI Conference on Analyzing Microtext*, pages 44–49. AAAI Press, 2011.

[217] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[218] Ehsan Hajiramezanali, Siamak Zamani Dadaneh, Alireza Karbalayghareh, Mingyuan Zhou, and Xiaoning Qian. Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9133–9142. Curran Associates Inc., 2018.

[219] Manu Sharma, Michael Holmes, Juan Santamaria, Arya Irani, Charles Isbell, and Ashwin Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 1041–1046. Morgan Kaufmann Publishers Inc., 2007.

[220] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, volume 2, pages 3320–3328. MIT Press, 2014.

[221] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58, 6 2020. ISSN 15662535. doi: 10.1016/j.inffus.2019.12.012.

[222] Salman Ul Hassan Dar and Tolga Çukur. A transfer-learning approach for accelerated MRI using deep neural networks. *CoRR*, abs/1710.02615, 2017. URL http://arxiv.org/abs/1710.02615.

[223] Saman Motamed, Isha Gujrathi, Dominik Deniffel, Anton Oentoro, Masoom A. Haider, and Farzad Khalvati. A transfer learning approach for automated segmentation of prostate whole gland and transition zone in diffusion weighted mri, 2019.

[224] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. URL https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[225] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: Generating images from limited data. In *ECCV*, 2018.

[226] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[227] Yaël Frégier and Jean-Baptiste Gouray. Mind2mind : transfer learning for gans. *CoRR*, abs/1906.11613, 2019. URL http://arxiv.org/abs/1906.11613.

[228] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv e-prints*, art. arXiv:1701.07875, Jan 2017.

[229] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, abs/1606.04797, 2016. URL http://arxiv.org/abs/1606.04797.

[230] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[231] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[232] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[233] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[234] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997. ISBN 0070428077.

[235] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[236] Selim Aksoy and Robert M Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern recognition letters*, 22(5):563–582, 2001.

[237] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31, 2012.

[238] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[239] Francesco Calimeri, Aldo Marzullo, Claudio Stamile, and Giorgio Terracina. Biomedical data augmentation using generative adversarial neural networks. In *Artificial Neural Networks and Machine Learning – ICANN 2017*, pages 626–634, 10 2017. ISBN 978-3-319-68611-0. doi: 10.1007/978-3-319-68612-7_71.

[240] Sahin Olut, Yusuf Huseyin Sahin, Ugur Demir, and Gozde Unal. Generative adversarial training for mra image synthesis using multi-contrast mri, 2018. URL https://arxiv.org/abs/1804.04366.

[241] Avi Ben-Cohen, Eyal Klang, Stephen P. Raskin, Shelly Soffer, Simona Ben-Haim, Eli Konen, Michal Marianne Amitai, and Hayit Greenspan. Cross-modality synthesis from ct to pet using fcn and gan networks for improved automated lesion detection, 2018. URL https://arxiv.org/abs/1802.07846.

[242] Andrew Ng. Gradient descent in practice I - feature scaling. Coursera. URL https://www.coursera.org/learn/machine-learning/lecture/xx3Da/gradient-descent-in-practice-i-feature-scaling.

[243] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[244] Martina Sollini, Lidija Antunovic, Arturo Chiti, and Margarita Kirienko. Towards clinical application of image mining: a systematic review on artificial intelligence and radiomics. *European journal of nuclear medicine and molecular imaging*, 46(13):2656–2672, 2019.

[245] Parnian Afshar, Arash Mohammadi, Konstantinos N Plataniotis, Anastasia Oikonomou, and Habib Benali. From handcrafted to deep-learning-based cancer radiomics: challenges and opportunities. *IEEE Signal Processing Magazine*, 36(4):132–160, 2019.

[246] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 7 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.3301590.

[247] Hieu H Pham, Tung T Le, Dat Q Tran, Dat T Ngo, and Ha Q Nguyen. Interpreting chest x-rays via cnns that exploit disease dependencies and uncertainty labels. *medRxiv*, 2019. doi: 10.1101/19013342. URL https://www.medrxiv.org/content/early/2019/11/29/19013342.

[248] Noemi Gozzi and Arturo Chiti. Explaining a xx century horse behaviour. *European journal of nuclear medicine and molecular imaging*, May 2021. ISSN 1619-7070. doi: 10.1007/s00259-021-05417-w. URL https://europepmc.org/articles/PMC8143985.

[249] Manuel Weber, David Kersting, Lale Umutlu, Michael Schäfers, Christoph Rischpler, Wolfgang Fendler, Irene Buvat, Ken Herrmann, and Robert Seifert. Just another "clever hans"? neural networks and fdg pet-ct to predict the outcome of patients with breast cancer. *European Journal of Nuclear Medicine and Molecular Imaging*, pages 1–10, 03 2021. doi: 10.1007/s00259-021-05270-x.

[250] Tensorflow. Tensorflow hub, 2019. URL https://tfhub.dev/. Accessed: 2021-04-29 10:33:03.

[251] Eli Gibson, Wenqi Li, Carole Sudre, Lucas Fidon, Dzhoshkun I. Shakir, Guotai Wang, Zach Eaton-Rosen, Robert Gray, Tom Doel, Yipeng Hu, Tom Whyntie, Parashkev Nachev, Marc Modat, Dean C. Barratt, Sébastien Ourselin, M. Jorge Cardoso, and Tom Vercauteren. Niftynet: a deep-learning platform for medical imaging. *Computer Methods and Programs in Biomedicine*, 2018. ISSN 0169-2607. doi: https://doi.org/10.

1016/j.cmpb.2018.01.025. URL https://www.sciencedirect.com/science/article/pii/S0169260717311823.

[252] MONAI Consortium. MONAI: Medical Open Network for AI, 3 2020. URL https://github.com/Project-MONAI/MONAI.

[253] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[254] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.