Memory-Efficient Deep Learning Methods for Brain Image Analysis

Inaugural dissertation

to
be awarded the degree of

Dr. sc. med.

presented at
the Faculty of Medicine
of the University of Basel

by
Florentin Bieder
from Langenbruck, Basel-Landschaft

Basel, 2024

Approved by the Faculty of Medicine
On application of

Prof. Dr. Philippe Claude Cattin, Universität Basel – *primary advisor*
Prof. Dr. med. Marios-Nikos Psychogios, Universität Basel – *secondary advisor*
Prof. Dr. Mattias Paul Heinrich, Universität zu Lübeck – *external expert*
Dr. Robin Sandkühler, Universität Basel – *further advisor*
Dr. Simon Pezold, Universität of Basel – *further advisor*

Basel, the 11th of November 2024

Prof. Dr. med. Dipl. phys. Eva Scheurer
Dean

Contents

Acronyms

ALARA as low as reasonably achievable. 18

API application programming interface. 33

BraTS brain tumor segmentation challenge. 10, 16, 17, 45, 63

CIFAR Canadian institute for advanced research. 23, 24

CNN convolutional neural network. 40

CPU central processing unit. 33, 34

CSF cerebrospinal fluid. 4

CT computed tomography. ix, 1, 2, 4, 10, 11, 13, 17–22, 40, 41, 93, 96

CUDA compute unified device architecture. 33, 34

DDIM denoising diffusion implicit models. 44, 48, 94

DDPM denoising diffusion probabilistic models. 48, 94

FLAIR fluid-attenuated inversion recovery. 10, 16, 48

GAN generative adversarial network. 39, 42, 45, 48, 95

GPU graphics processing unit. ix, 2, 33–35, 93, 94

HD95 95th-percentile Hausdorff distance. 47

HDD hard disk drive. 34

HGG high grade glioma. 8, 10

HU Hounsfield unit. 21

IEEE institute of electrical and electronics engineers. 34

INR implicit neural representation. ix, 2, 40–42, 45, 95

LGG low grade glioma. 8, 10

LLM large language model. 40

MICCAI medical image computing and computer-assisted intervention. 81

MIDL conference on medical image deep learning. 49, 63

MLP multilayer perceptron. 25, 27, 28, 33, 41

MR magnetic resonance. ix, 1, 4, 9, 10, 15, 16, 41, 45, 96

MRI magnetic resonance imaging. ix, 1, 2, 8, 9, 13, 16, 17, 22, 23, 41, 47, 48, 93, 95

MS multiple sclerosis. 48

MSE mean squared error. 29, 46

NeRF neural radiance fields. 40

NHS national health service. 1

RAM random access memory. 34

ReLU rectified linear unit. 25, 29, 41

RF radio frequency. 14–16

SGD stochastic gradient descent. 30

SIREN sinusoidal representation networks. 41

SM streaming multiprocessor. 34, 35

SSD solid-state drive. 34

TBI traumatic brain injury. 10

VAE variational autoencoder. 39

VRAM video-RAM. 34, 35

WHO World Health Organization. 8

XOR exclusive-OR. 25, 28

Acknowledgements

First, I want to thank Philippe C. Cattin. He gave me the opportunity to work in his group, for which I'm very grateful. He continuously encouraged me to explore new ideas and directions, even when unsure about their potential or likelihood of success. While certainly not all of them worked out, these experiences taught me so much about a wide variety of topics. Then, I would like to thank my two supervisors: Simon Pezold, who enabled my project in the first place and equipped me with all the support and tools I needed to start this journey and Robin Sandkühler, who always inspired me to explore more and dive deeper. He challenged me in many discussions and shared his enthusiasm for the field we work in but also for things entirely outside of the field. I am also very grateful for the support of Julia Wolleb, without whom this work would not have been possible. For the whole journey, she was my partner-in-crime and good friend. We were rarely separated by more than two computer screens, and I could always rely on her and talk to her about *all* my problems. Furthermore, I would like to thank Alina Giger, Antal Huck-Horváth, Christoph Jud, Peter von Niederhäusern, Eva Schnider, Carlo Seppi and Bruno Sempéré, who served me as role models and mentors, especially at the start of my journey. I also very much appreciated the help and support of Hélène Corbaz, Alicia Durrer, Paul Friedrich and Philippe Valmaggia, with whom I've had frequent discussions and who gave me a lot of valuable feedback for my papers and my thesis. I spent an amazing time in this group and would like to thank all its present and past members for their friendship and support: Annette Mettler, Arnaud Champetier, Balázs Faludi, Florian Spiess, Lorenzo Iafolla, Juval Gutknecht, Karim Ayadi, Madina Kojanazarova, Marek Zelechowski, Massimiliano Filipozzi, Michael Wehrli, Moira Zuber, Nair von Mühlenen, Natalia Mañas Chavernas Negin Sahraei, Norbert Zentai, Oumeymah Cherkaoui, Peter Zhang, Samaneh Manavi, Sergei Pnev, Susanne Schaub, Simon Fluder, Tamás Faludi, Tiziano Ronchetti Vincent Ochs, Volodimir Buchakchiysky, and also all the other members of the department, who are too many to list here. I would, however, like to highlight everyone at our department whose work often goes unnoticed, which includes Angela Duarte Beat Fasel, Beat Göpfert, Constanze Pfeiffer, Corinne Eymann-Baier, Daniela Vavrecka-Sidler, Gabriela Oser Duss, Hannah Heissler, Sara Freund. Finally, I would like to thank my family and friends for enabling me to do these studies and who have supported me since the beginning.

This project was financially supported by the Novartis FreeNovation initiative and

the Uniscientia Foundation.

Summary

The diverse array of imaging modalities currently in use has profoundly impacted the practice of medicine. In particular, X-ray computed tomography (CT) and magnetic resonance imaging (MRI) have the capacity to record a three-dimensional image of the body with great detail. As a consequence of their widespread use, the amount of available data is continuously increasing at an unprecedented rate. This presents an opportunity to apply machine learning and, in particular, deep learning. Deep learning is a subfield of machine learning that is particularly well-suited to train on large quantities of data and to capture complex features in high-dimensional data. Currently, deep learning methods rely on graphics processing units (GPUs) to be evaluated with any reasonable speed. However, GPUs with high-performance characteristics – which are directly linked to their memory capacity – contribute significantly to the high implementation costs of deep learning methods. To enable an easier access and a more widespread adoption, it is crucial to reduce this barrier of entry.

In this work, we explore the potential of deep learning methods for a range of tasks involving brain imaging data, with a focus on reducing the GPU-memory consumption. This is particularly relevant for methods that process MR and CT scans, as they inherently are three-dimensional. The first project we present is an unsupervised anomaly detection and localization method for brain CT scans. It employs a novel self-supervised surrogate task. The second project involves a supervised tumor segmentation in MR scans with denoising diffusion models. This method has previously been proposed for two-dimensional slices. As diffusion models already require substantial resources for two-dimensional data, we explore various techniques to reduce the resource consumption, to adapt this model to three-dimensional data. In the last project, we explore implicit neural representations (INRs) to model the development of the neonatal brain on the basis of MR scans. We show that INRs can be trained in sparsely sampled data, explore techniques for disentangling the latent space and illustrate how they can be trained with minimal resources.

Zusammenfassung

Die vielfältigen bildgebenden Verfahren, die heute im Einsatz sind, haben die Medizin revolutioniert. Insbesondere Computertomografie (CT) und Magnetresonanztomografie (MRT) können verwendet werden, um dreidimensionale Bilder des Körperinneren detailliert zu erfassen. Infolge ihrer weitverbreiteten Nutzung stehen heute so viele Daten zur Verfügung wie nie zuvor. Dies bietet eine Gelegenheit, maschinelles Lernen und insbesondere Deep Learning einzusetzen. Deep Learning ist ein Teilbereich des maschinellen Lernens, das sich besonders gut für das Training basierend auf grossen Datenmengen eignet, und besonders gut komplexe Features in hochdimensionalen Daten erfassen kann. Derzeit sind Methoden des Deep Learnings auf performante Grafikprozessoren (GPUs) angewiesen. Allerdings können GPUs mit hohen Leistungsmerkmalen – welche direkt mit der Grösse ihres Speichers im Zusammenhang stehen – erheblich zu den Kosten beitragen, die mit der Nutzung von Deep Learning Methoden verbunden sind. Um eine breitere und einfacher zugängliche Verwendung zu ermöglichen, ist es entscheidend, diese Eintrittsschwelle zu senken. In dieser Arbeit untersuchen wir das Potenzial von Deep Learning Methoden für eine Reihe von Aufgaben mit bildgebenden Daten des menschlichen Gehirns, mit dem Schwerpunkt auf der Reduktion des Bedarfs an Speicher der GPUs. Dies ist besonders relevant für Methoden, die MR- und CT-Bilder verarbeiten, da diese von Natur aus dreidimensional sind.

Das erste Projekt, das wir vorstellen, handelt von einer unüberwachten Methode zur Anomalieerkennungs und -lokalisierung für CT-Bilder des Gehirns. Diese Methode verwendet eine neue, self-supervised Surrogate-Aufgabe vor. Das zweite Projekt dreht sich um eine überwachte Hirntumorsegmentierungsmethode für MR-Bilder mittels Denoising Diffusion Models, die zuvor für zweidimensionale Schnittbilder entwickelt worden ist. Da Diffusion Models bereits für zweidimensionale Daten erhebliche Mengen an Speicher benötigen, untersuchen wir verschiedene Techniken um mit dieser Schwierigkeit umzugehen, um sie schlussendlich für dreidimensionale Daten anwenden zu können. Im letzten Projekt untersuchen wir Implicit Neural Representations (INR), die auf dünn gesampelten Daten trainiert werden können. Wir untersuchen Techniken für das Disentanglement des Latent Space, und für das Training minimalen Ressourcen.

Chapter 1

Introduction

In 1895, Willhelm C. Röntgen discovered X-rays, which marked the dawn of medical imaging. For the first time, it was possible to look through the human body without cutting it open. Up to this day, it remains one of the most common imaging modalities used for diagnostics. During the First World War, Marie Skłodowska Curie built cars equipped with X-ray systems – later called "petite Curies" – which were the first *mobile* imaging system [1]. Soon after, the contrast agents for imaging the gastrointestinal tract were discovered, which again opened up many new opportunities [2]. The X-rays soon gave rise to the *focal plane tomography* (or *planigraphy*), developed during the 1920ies and 1930ies [3]. This technique consists of exposing an X-ray film several times from different angles, which had the effect that only one specific plane inside the body remained in focus. In that sense, it was the first *tomographic* modality. With the advent of electronic computers, the possibilities grew, and in the 1970ies, ultrasound imaging, magnetic resonance imaging (MRI) and X-ray computed tomography (CT) were used clinically for the first time [2]. Since then, many new techniques and applications have been developed, and they have become faster, and their imaging resolution increased. The English national health service (NHS) reports that between 2013 and 2023, the number of MRIs performed grew by over 70% and the number of CT scans more than doubled [4].

The fast growth of the amount of medical imaging that is being performed also poses challenges for the healthcare systems. This increases healthcare costs and the burden on the radiology staff [5]. Due to the amount of data that is being recorded, it is of interest to use automated image analysis. This is a great opportunity to apply machine learnig, as it can make use of large amounts of data. Therefore, it has the potential to lighten the burden on the radiology staff and it can enable or accelerate studies involving larger cohorts.

1.1 Motivation

In this work, we focus on the brain imaging data, CT- and MR-scans specifically. Due to their relatively high resolution, and because of their three-dimensional nature, any

implementation of processing algorithms require careful consideration of the needed resources.

Deep learning models can have particularly high requirements regarding the computational resources and profit from a highly parallelized graphics processing units (GPUs). However, GPUs incur a significant cost, which can limit the use of deep learning models in practice. Especially the GPUs memory is one of the main bottlenecks when applying deep learning in medical image analysis. We discuss this in more detail in Section 4.2.6. The three projects presented in this thesis explore methods that inherently do not require a lot of GPU memory or that can be optimized to reduce the memory consumption.

1.2 Contribution

We discuss three projects involving different brain image processing tasks on CTs and MRIs scans. We developed a novel unsupervised anomaly detection and localization method based on patch-wise position regression as a surrogate task with a self-supervised training. The patch-based approach makes use of relatively small networks. We present it in Chapter 6.

The second project is about a brain tumor segmentation method using denoising diffusion models, which has previously been proposed for two-dimensional data. As diffusion models are generally resource-intensive, we propose multiple technical contributions to enable this method for three-dimensional data. This project is presented in Chapter 7.

In the third project, we explore the use of implicit neural representations (INRs) as an approach for modeling the neonatal brain development over time. We show how INRs can efficiently be trained with limited resources and evaluated on high-dimensional data, and propose two methods that encourage the age-disentanglement within the INRs latent space. We present this project in Chapter 8.

1.3 Outline

Chapter 2 gives a short overview over the brain anatomy in Section 2.1 and its early development in Section 2.2. Furthermore, we discuss injuries and disorders relevant to these projects, specifically gliomas in Section 2.3, and intracranial hemorrhages and cranial fractures in Section 2.4. In Chapter 3, we introduce the imaging modalities that were used to acquire the datasets we have used in this project. They comprise magnetic resonance imaging in Section 3.1 and X-ray computed tomography in Section 3.2. Because all three projects involve the development of deep learning methods, we introduce the foundational concepts of deep learning in Chapter 4, and provide more in-depth context for our methods in Chapter 5. Finally, chapters 6 to 8 include the publications of these projects, with a discussion in Chapter 9.

Chapter 2

Clinical Background

In this chapter, we discuss the imaging techniques used for the data we analyzed in the projects presented in Chapter 6, Chapter 7, and Chapter 8, as well as the problem settings and the underlying pathologies.

2.1 Overview of Skull and Brain Anatomy

The human brain is a component of the central nervous system, which also includes the spinal cord. It controls the motor function of the body, interprets sensory information, and regulates human behavior. The brain is located within the cranial cavity of the skull. The bones that surround the cranial cavity are collectively referred to as the neurocranium, while the remaining bones of the skull are known as the facial bones. The neurocranium, shown in Figure 2.1, consists of the os occipitale, the os parietale (dextrum et sinistrum), the os frontale, the os temporale (dextrum et sinistrum), and the os sphenoidale. The calvaria is the superior part of the neurocranium and consists of the pairs of frontal and parietal bones, as well as the superior parts of the occipital bone and the temporal bones [6].

Macroscopically, the brain can be divided into four parts, namely the brainstem, the cerebrum, the diencephalon and the cerebellum. The brainstem extends the spinal cord and comprises the mesencephalon, pons, and medulla. The cerebrum is the largest of the four main parts, and can again be separated into the left and right hemispheres. The outermost layer of neural tissue of the cerebrum is the cerebral cortex, exhibiting the characteristic folds and grooves of the human brain, called *gyri* and *sulci*. The diencephalon is in the center of the cerebrum and superior to the brainstem. The cerebellum is in the anterior part of the cranial cavity and below the cerebrum.

As illustrated in Figure 2.2, the brain and the remainder of the central nervous system are encased in three meninges. The outermost layer is the dura mater, followed by the arachnoid mater, and then the pia mater. In the skull, the dura mater is typically attached to the bone and is in direct contact with the arachnoid mater. It has two notable extensions into the cranial cavity. On the one hand, the falx cerebri extends into the longitudinal fissure, separating the two cerebral hemispheres. On the other hand, the

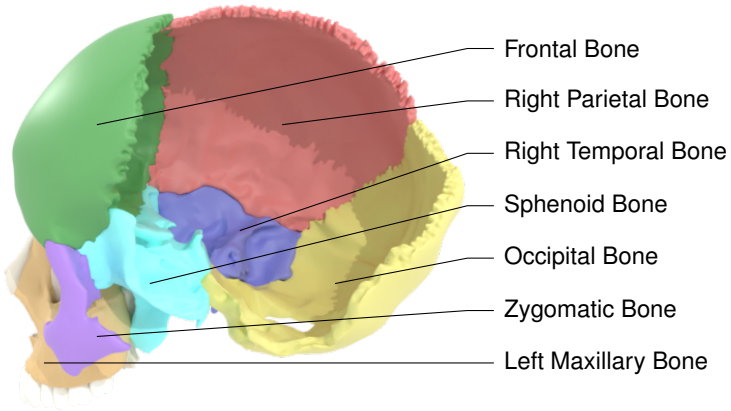


Figure 2.1: The Skull bones, with the left temporal and parietal bone removed. Data visualized from BodyParts3D [7].

tentorium cerebelli separates the cerebrum from the cerebellum [8]. The pia mater closely follows the surface of the brain and extends into the sulci. The subarachnoid space, i.e., the space between the arachnoid mater and the pia mater, is filled with the cerebrospinal fluid (CSF) [9]. The CSF surrounds the whole central nervous system and also occupies the space in the ventricles of the brain. It has a cushioning function, supplies nutrients, and removes metabolites from the central nervous system [10].

The brain parenchyma can be divided into grey and white matter. The grey matter contains primarily the cell bodies of the neurons, while the white matter mostly consists of the myelinated axons that extend from the cell cores and connect to other neurons [8]. The *neurons* are the cells of the nervous system. They are responsible for transmitting electrical signals across the body [12]. Their cell body has dendrites that extend to other neurons to receive signals, and an axon, which can be a long process extending from the cell body, ending in the synaptic terminals that can release neurotransmitters and therefore transmit signals to other cells. As mentioned above, the cerebral cortex is the cerebrum's superficial layer, consisting of grey matter. The difference between white and grey matter is visible in CT as well as in some MR sequences, as we will further discuss in Section 3.

Furthermore, the brain parenchyma contains glial cells, providing physiological support to the neurons [13, 14]. The glial cells consist mainly of astrocytes, oligodendrocytes, ependymal cells and microglia [14]. Astrocytes form the largest group of the glial cells and have various functions, including structural support, regulation of the metabolism and repair of damage in the tissue [15]. Oligodendrocytes are responsible for the myelination of the axons [16]. The ependymal cells line the ventricles and the choroid plexus as the interface between the brain parenchyma and the CSF. They are

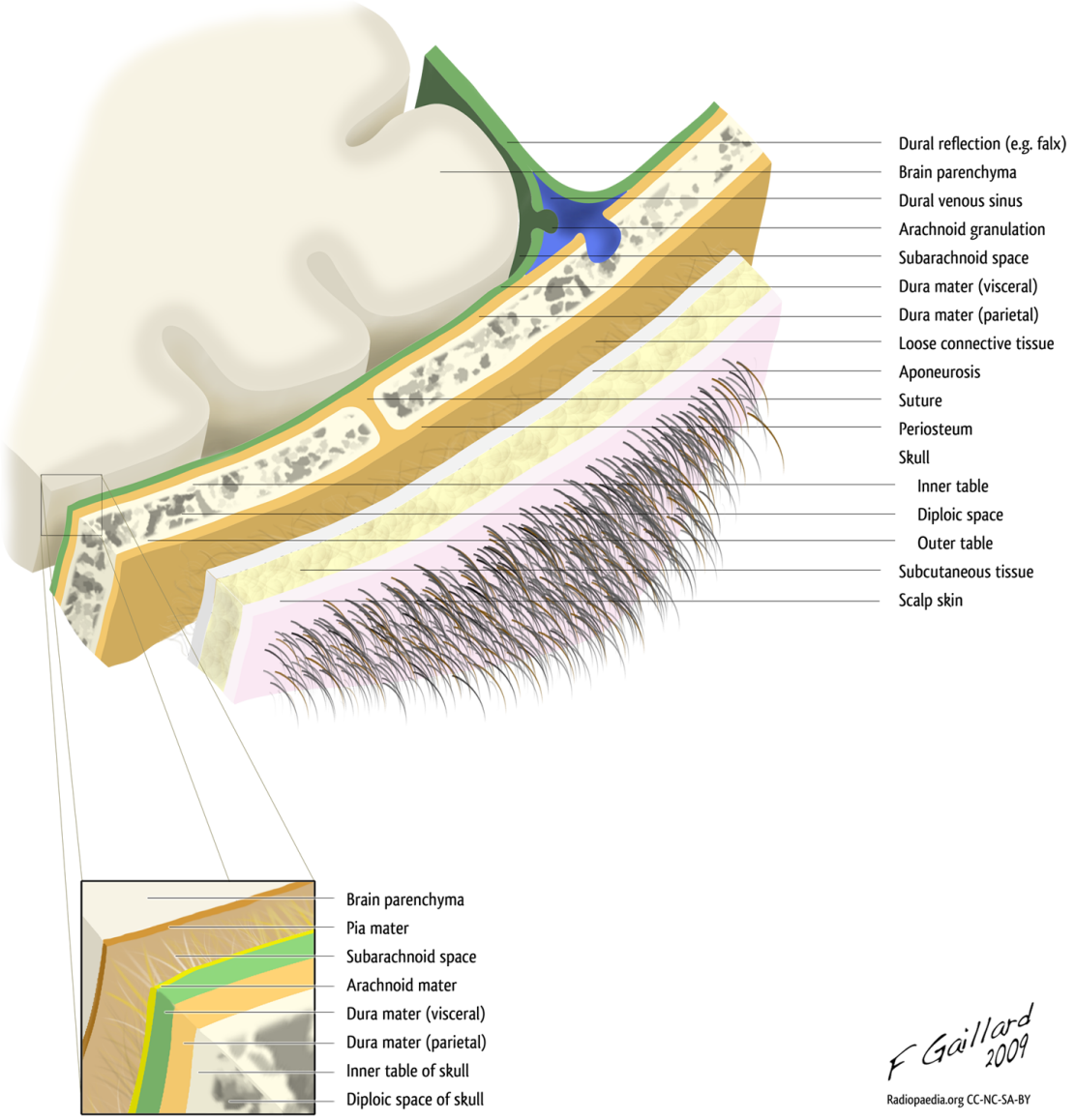


Figure 2.2: Illustration of three meninges between the brain and the skull. Courtesy of Frank Gaillard [11].

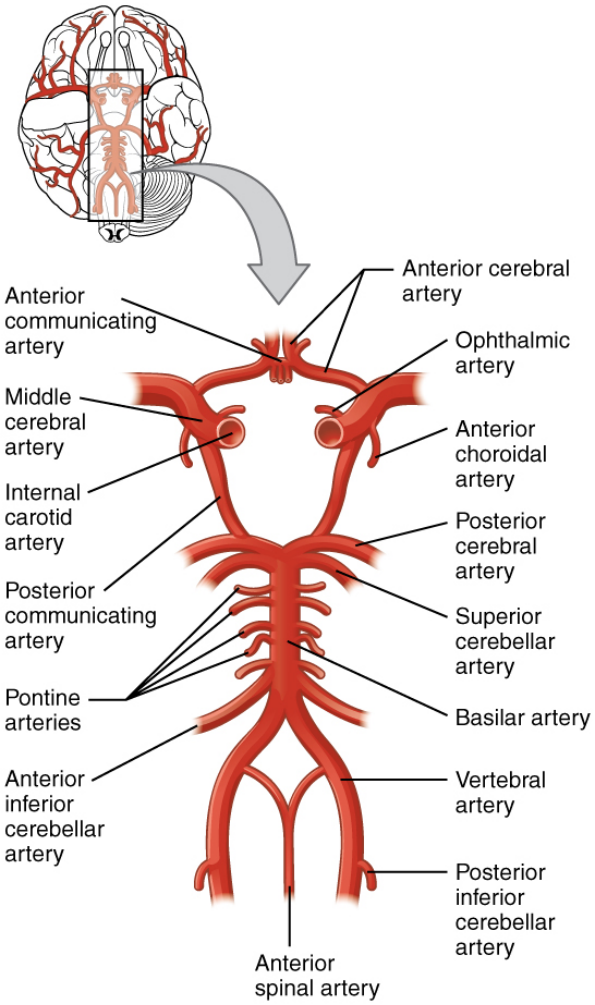


Figure 2.3: Illustration of the arterial supply of the brain. Courtesy of OpenStax [19].

responsible for the production and the exchange of CSF [13]. Finally, the microglia are part of the macrophage system and are, therefore, part of the immune system [17].

The blood is primarily supplied to the brain via the internal carotid arteries and the vertebral arteries. The arterial system of the brain has multiple redundant paths [18] as shown in Figure 2.3.

The vertebral arteries join each other again to form the basilar artery. The left and right posterior cerebral arteries branch off the basilar artery and supply the posterior brain. From each of the internal carotid arteries, the corresponding middle cerebral arteries and anterior cerebral arteries branch off. The two anterior cerebral arteries are connected by the anterior communicating artery, and the internal carotid arteries are each connected to the posterior cerebral arteries via posterior communicating arteries [19]. From that point, the major vessels branch out while traversing the subdural

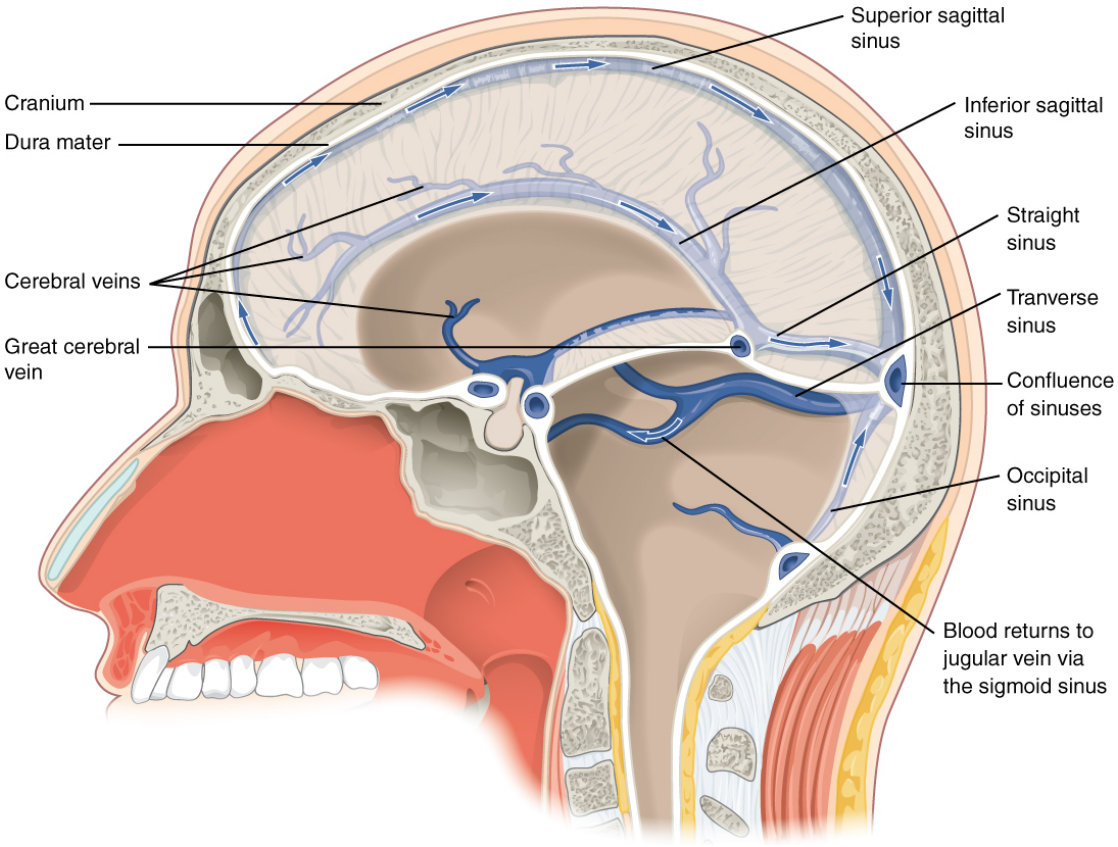


Figure 2.4: Illustration of the dural venuses. Courtesy of OpenStax [19].

space. Subsequently, multiple finer vessels enter into the brain parenchyma. The vessels exiting the parenchyma return the blood through the veins that occupy the dural sinuses, as shown in Figure 2.4.

2.2 Fetal and Neonatal Development of the Brain

This section aims to provide an overview of the development of the fetal and neonatal brain. For this period of the development, there are multiple conventions for defining the *age* of the subject. It is important to distinguish them and thus, we first want to clarify them. In this section, we use two conventions: The first is the age *post conception*, which is the preferred terminology for discussing development, mainly during the embryonic and fetal phase. The second convention comprises two terms, the *gestational age* and *postmenstrual age*. Both of them refer to the age since the first day of the last menstrual period, but the former is exclusively used during pregnancy and to indicate the length of the pregnancy, while the latter is the preferred term for the perinatal period [20]. By convention, if the date of conception is known (e.g., due to assistive

reproductive procedures), the gestational age is computed by adding two weeks to the age post conception [20].

First Trimester In the third week post-conception, the embryo consists of three germ layers, specifically the ectoderm, the mesoderm, and the endoderm [18]. The central nervous system arises from the ectoderm, the outermost layer. It develops the neural plate, which then shapes into the neural tube, which will contain the cerebrospinal fluid. This neural tube develops into the forebrain, midbrain and hindbrain after about four weeks [21]. The forebrain later develops into the telencephalon, which in the adult brain becomes the cerebrum, and the diencephalon, which in the adult brain includes notably the thalamus and hypothalamus but also the retina [18]. The hindbrain develops into the pons and medulla oblongata, forming the brainstem, the midbrain, and the cerebellum. By week eight, the internal capsule is present, and the cortical plate begins to form along with other major structures. Furthermore, the major organs are also present, marking the end of the embryonic phase [21]. By the twelfth week, the cerebral cortex has undergone significant growth, and the two hemispheres have been connected by the corpus callosum, which has begun to form. [21].

Second Trimester The second trimester is approximately between week 13 and 26 post-conception. The number of synapses starts to increase rapidly at the beginning of the second trimester. The myelination of the neurons begins at around 20 weeks and continues until several years after birth. The vascularization of the brain starts to develop between 22 and 24 weeks post conception [21].

Third Trimester The third trimester lasts from about week 27 post-conception up to birth. During the end of the second and the beginning of the third trimester, the cerebral cortex's growth speed increases. Due to its growth, the gyrification also advances rapidly. While the lateral-, the parieto-occipital-, and the central sulci are already developed by the end of the second semester, most of the major sulci are formed in the window specifically between week 26 and week 30 [22]. Afterward, until well after birth, the gyrification continues. We display brain MRIs of neonates in this age range in Figure 2.5.

2.3 Gliomas

Various types of brain tumors can be classified by their origin. Gliomas specifically originate from glial cells. As described in Section 2.1, these are cells in the central nervous system that do not transmit information, but among many other tasks, regulate the composition of the extracellular space and also make up the blood-brain barrier [8]. Based on their malignancy, the World Health Organization (WHO) classifies gliomas into four grades (I - IV) [13, 25]. The grades I and II encompass the *low grade gliomas* (LGGs) that are benign. The grades III and IV are called the *high grade gliomas* (high grade gliomas) and include the malignant glioblastomas (grade IV) [25]. Glioblastomas and

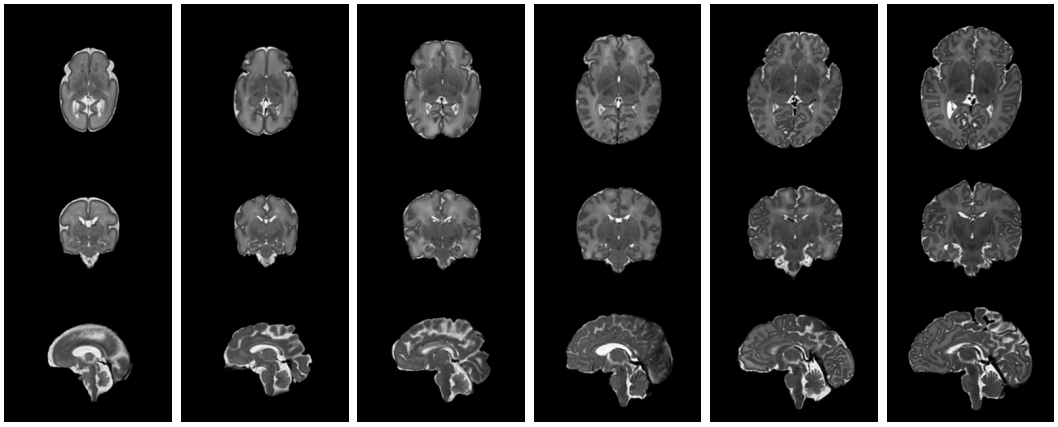


Figure 2.5: Brain MRIs of preterm and term neonates at 27-42 weeks postmenstrual age. We show each an axial, coronal and sagittal slice of T_2-weighted images. Data visualized from the developing Human Connectome Project [23, 24].

astrocytomas are both astrocytic, i.e., tumors originating from astrocytes, but usually, only grades I-III are called astrocytomas while grade IV are glioblastomas [13].

In general, brain tumors are rare, with an incidence of 5-13 per 100000 per year [26]. The most common ones are, however, the malignant glioblastoma: According to a study in England [27] the most common brain tumors are glioblastoma (32%), followed by meningoma (27%), other astrocytoma (9%), oligodendroglioma (3%), lymphoma (2%), ependymoma (1%). Note that all of those types except for meningoma and lymphoma are glioma. With magnetic resonance imaging, it is possible to distinguish different parts of a glioma [25]:

- the GD-enhancing tumor: In MR-imaging, frequently gadolinium-based contrast agents are used. The part of the tumor appearing hyper-intense in T1-weighted images is called the *enhancing tumor*, as the contrast agent *enhances* said part in this MR-sequence. This is mostly visible in Glioblastoma, but usually less visible in low-grade-gliomas, as the blood-brain barrier is disrupted to a lesser degree.
- the peritumoral edema: Some parts of the surrounding tissue can also be invaded by the tumor, resulting in edema, typically visible as hyper-intense regions in T_2-weighted images.
- the necrotic and non-enhancing tumor core: Within the enhancing rim of the tumor, a necrotic core can appear.

The treatment options depend on the type, location, size, and severity of the tumors, as well as the preferences of the patient. The typical options are active surveillance, surgical resection, radiation therapy, chemotherapy and targeted therapy [28]. The segmentation of brain tumors can be used as a step for the diagnosis as well as the treatment planning. For example, for radiotherapy, the radiation oncologist needs

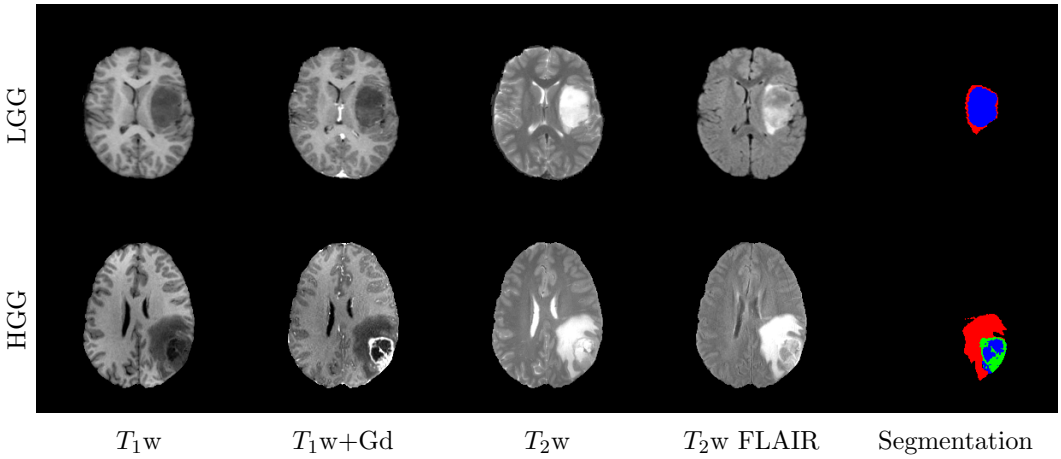


Figure 2.6: An example of a high- and low-grade glioma (HGG, LGG) along with three delineated segments: red: the peritumoral edema, green: the gadollinium-enhancing part, blue: the necrotic and non-enhancing tumor core. Four different MR images of the same subjects are shown: A T_1-weighted image, a T_1-weighted image after injection of the Gadollinium-based contrast agent, a T_2-weighted image as well as a fluid-attenuated inversion recovery (FLAIR) sequence. Data visualized from BraTS2020 [25].

to define a three-dimensional volume that needs to be treated with a certain dose of radiation [29].

2.4 Intracranial Hemorrhages and Cranial Fractures

Head injuries can have severe consequences, and it is important to detect them early and treat them appropriately. In this section, we focus on calvarial fractures and intracranial hemorrhages. Traumatic brain injuries (TBIs) consist of injuries to the brain by external forces, and can be divided into two categories, namely *closed* head injuries, including blunt trauma and blast injuries, and *penetrating* injuries, including gun shot and stabbing [30]. Both of them can involve fractures of the skull, notably calvarial fractures, as well as intracranial hemorrhaging. For traumatic injuries of the head, a CT scan is the recommended diagnostic tool [31]. Non-contrast CT is suitable to detect fractures of the skull as well as intracranial hemorrhages. Figure 2.7 shows a non-contrast CT of a patient with extradural hemorrhage and a calvarial fracture.

2.4.1 Cranial Fractures

Fractures of the skull are classified into three categories, i.e., mandibular, basilar, and calvarial fractures. They are caused by high mechanical stresses, e.g., impacts from accidents or violence [32]. Furthermore, they can be classified, on the one hand, into closed and open fractures, the latter of which have an elevated risk of infection. On

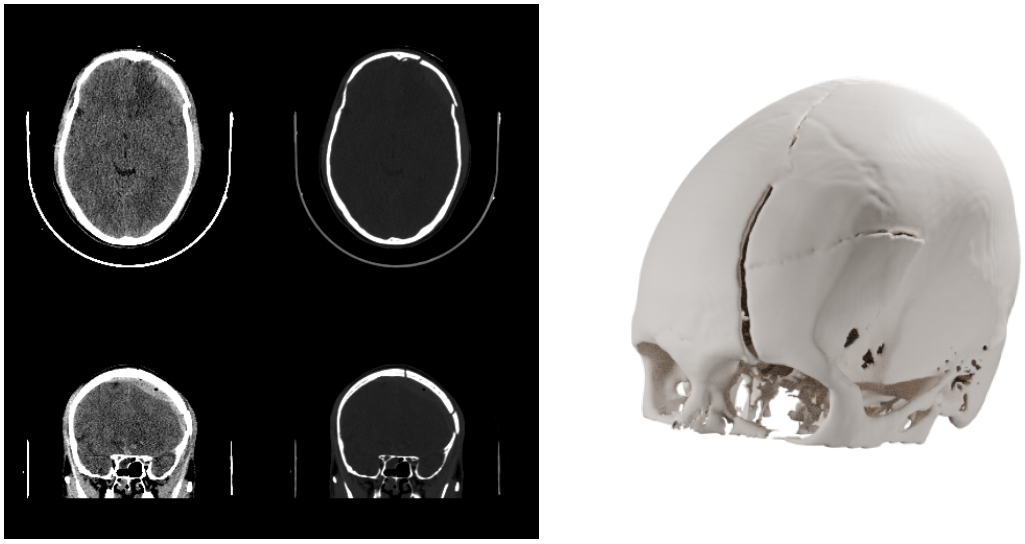


Figure 2.7: A CT of a case of a calvarial fracture with extradural hemorrhage, shown as axial and coronal slices windows appropriate for soft tissue and bone, and as volume rendering, Data visualized from CQ500 dataset [33].

the other hand, they can be classified into two categories that distinguish whether the fragments have been displaced (often depressed, i.e., displaced into the skull). If fractures are suspected, volumetric CT scans are performed. This allows the physicians to clearly identify the fracture lines in any direction, and whether multiple fragments are involved [32].

2.4.2 Intracranial hemorrhage

Common risks for intracranial hemorrhage are hypertension, cerebral amyloid angiopathy (i.e., the accumulation of amyloid proteins in the vessel walls weakening their structure), coagulopathy (i.e., an impairment of the blood to coagulate), structural lesions (e.g., arteriovenous malformations, brain tumors and aneurisma), drug abuse and prior ischemic strokes [34]. We can distinguish different types of hemorrhages depending on their location with respect to the different tissues, especially the meninges [35]: The first coarse distinction is between extra-axial and intra-axial hemorrhages. The former comprises epidural and subdural ones, while the latter comprises subarachnoidal, intraparyncheal and intraventricular hemorrhages. The intraventricular hemorrhages are sometimes understood to be a type of the intraparyncheal hemorrhages [36]. All of these can be caused by non-penetrating and penetrating trauma.

When an intracranial hemorrhage is suspected, the method of choice is usually a CT scan. In the native CT (i.e., without contrast enhancement), the blood appears hyperdense and is visually easily perceptible, provided there is a sufficient amount of blood [36]. The treatment can include blood pressure control to avoid hypertension, reverse coagulopathy and surgical evacuation in sufficiently severe cases [37].

Chapter 3

Imaging Modalities

In this chapter, we want to focus on two imaging modalities that are pervasive in medicine: Magnetic resonance imaging (MRI) and X-ray computed tomography (CT). Both of them are forms of *tomography*. Tomography means that it allows us to virtually slice through the object of interest, that is, it let us look into the inside of a body without actually cutting it open.

3.1 Magnetic Resonance Imaging

Magnetic resonance imaging is a non-invasive tomographic imaging modality that does not rely on ionizing radiation. It allows us to differentiate different types of soft tissues very well and at a high resolution, which enables its use for the diagnosis of a wide variety of conditions.

3.1.1 Nuclear Magnetic Resonance

Hydrogen Nuclei, i.e., protons, possess an intrinsic quantum mechanical property called *spin*, Spin is frequently compared to angular momentum, as a helpful analogy [38], because it behaves similarly in the presence of an electric field as angular momentum does in the presence of a gravitational field: We can think of the spin as the rotation of the charge around its own axis, creating a magnetic field, even though there is no physical rotation [39]. Without any external influences, the orientation stays the same. In contrast to angular momentum, spin is quantized. In the presence of a magnetic field B_0, the protons attain two different energy levels (*spin up* and *spin down*), which is called the *Zeeman splitting*. They align with the magnetic field in that they point in the same or in the exact opposite direction. In this case, the number of *spin up* protons $N\uparrow$ is only slightly greater than the number of *spin down* $N\downarrow$ protons. The distribution can be described by the Boltzmann distribution and can also be expressed as the *polarization* p [40], i.e., the relative amount of protons contributing to the net magnetization:

$$p = \frac{N\uparrow - N\downarrow}{N\uparrow + N\downarrow} \approx \frac{\gamma h B_0}{4\pi k_B T} \tag{3.1}$$

where T is the temperature k_B is the Boltzmann constant, and h the Planck constant. For water at room temperature in an 1.5 T field, this is roughly 10^{-5}, which means a large part of the magnetic moments cancels out.

If the magnetic field changes direction, the magnetization will start to precess, just like when a torque is applied to a spinning object with some angular momentum [40]. This precession causes the emission of an electromagnetic wave that we can measure. As energy is slowly lost due to effects we will discuss later, the precession decays until the spin is again aligned with the magnetic field. The frequency of the precession is described by the Larmor equation

$$\omega_0 = \gamma B_0 \tag{3.2}$$

where ω_0 is the *Larmor frequency*, i.e., the angular frequency of the precession, B_0 the magnetic field and γ the *gyromagnetic ratio*, which is a property of the type of nucleus. Alternatively, we can also use an electromagnetic signal to *excite* the spin, i.e., turn it away from the magnetic field, to make it precess [38]. This signal is called a *radiofrequeny*-Impulse (RF), which is a signal at Larmor frequency. It allows for flipping the spins by any angle α. For simplicity, we assume an angle of $\alpha = 90°$. The magnetization of a single spin can always be described using a unit vector in a Cartesian coordinate system. For imaging, we are interested in the net magnetization M of a whole collection of protons. Outside of a magnetic field, the spins would point in random directions and cancel each other out, i.e., the net magnetization would be $|M| = 0$. Let us use a Cartesian coordinate system and use the magnetic field B_0 as our z direction. Since the spins precess around the z-axis (i.e., within an xy-plane), it is conducive to split the net magnetization into a z-component M_z and the xy-component M_{xy}, as is also shown in Figure 3.1.

3.1.2 T_1- and T_2-Decay and Radiofrequency Pulses

With an RF-signal, it is possible to move the precession into the xy-plane, such that $|M_{xy}| = M_0$ and $M_z = 0$. However, the magnetization will soon after return to the initial state. During this transition, we can observe two independent effects: The *spin-spin-relaxation* (or T_2-decay) and the *spin-lattice-relaxation* (or T_1-decay) [40].

Within a short time after the RF-excitation, the transverse magnetization $|M_{xy}|$ will decay to zero, this is called the *spin-spin-relaxation*. The reason for this is the dephasing of the precession of the collection of protons we are considering are caused by the interaction of other close-by protons. These interactions induce slight variations in the magnetic field and therefore slight changes in the Larmor frequency. Therefore, the energy is being redistributed among the protons. This effect is largely independent of the magnetic field B_0 [39]. This process is accelerated proportionally to the inhomogeneities caused by the scanner itself as well as the scanned object, resulting in a shorter decay time T_2^* with

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \gamma \Delta B_0 \tag{3.3}$$

where ΔB_0 is the variation in the magnetic field [41]. Furthermore, in a slower process

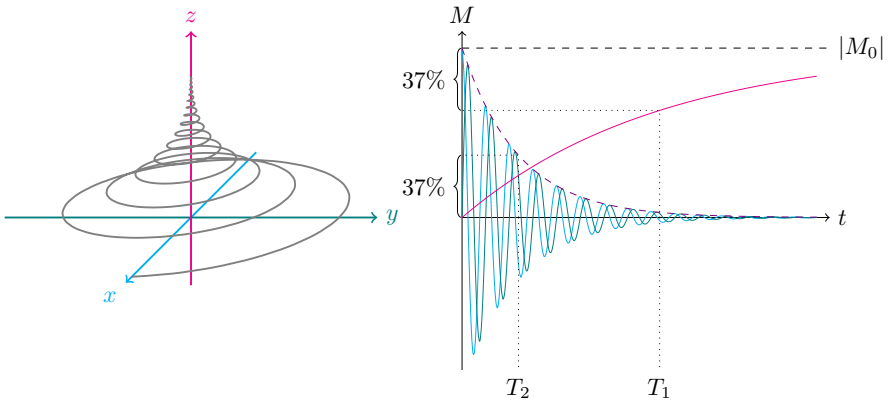


Figure 3.1: The trajectory of the magnetization vector M over time according to Equation (3.4) after a 90° RF-pulse, tipping the magnetization into the xy-plane. The $37\% \approx e^{-1}$ marks the decay after the times T_1 and T_2 respectively.

generally, the longitudinal magnetization $|M_{xy}|$ will converge to M_0. This happens due to the loss of energy, which is transferred to the surrounding tissue (i.e., the *lattice*). It is dependent on B_0 but also the movement of the surrounding molecules [39].

After the RF-excitation at $t = 0$, we can model these two decay types using

$$M_{xy} = M_0 \exp(-t/T_2) \begin{pmatrix} \cos(\omega_0 t) \\ \sin(\omega_0 t) \end{pmatrix} \text{ and } M_z = M_0(1 - \exp(-t/T_1)), \tag{3.4}$$

where T_1 and T_2 are the characteristic decay times for the two processes. This process is schematically shown in Figure 3.1.

3.1.3 MR-Scanners

An MR-scanner consists of multiple components. An electromagnet provides the magnetic field B_0. In clinical scanners, they can be resistive or superconducting [39], along with additional smaller electromagnets that allow to tune the homogeneity of the field precisely, called *shimming magnets*. The most common form is a toroidal magnet with its central opening horizontally, to be accessed by the patient table. Furthermore, there are *gradient coils*, i.e., electromagnets that can precisely introduce gradients into the magnetic field during a scan. Then there is an antenna, also in the form of a coil around the central bore, that is used for the transmission of the required RF signals to excite and steer the spins. To record the signals, smaller antennas are placed close to the body part to be scanned. In addition, supporting equipment like the RF transmitter, receiver and amplifiers, a cooling system, RF- and magnetic shielding for the room containing the scanner is needed [40].

3.1.4 MR-Sequences

There exists a vast amount of so-called (MR-) sequences for all kinds of different purposes. They define a precise sequence of RF-pulses, gradients and recording windows. The research in the design of such sequences is still ongoing and driven by new research questions [42].

Many of the sequences used for imaging can be divided into T_1- or T_2-weighted images, depending on which of those effects mainly contributes to the contrast.

For imaging, we need to relate the signals that are emitted from the protons with locations in our volume of interest. This can be achieved using *spatial encoding*. It is an important part of imaging sequences and involves different techniques that can be combined in many ways. A simple example is the *slice selection*. If we want to image a specific plane, we can apply a gradient to the magnetic field along the direction perpendicular to the desired plane. This has the effect that only the spins within that plane will be excited by the RF-pulse, as due to the gradient in the magnetic field, all protons outside of said plane have a slightly different Larmor frequency. Furthermore, *phase-* and *frequency-encoding* are used to produce signals of specific parts of the *k-space*, which we will introduce below. For a complete explanation of the spatial encoding and MR-sequences in general, we would like to refer to [42]. In many common sequences, the contrast in the images comes, therefore primarily from the different relaxation times of different tissues. It is also possible to image other properties, e.g., *proton density* weighted images, where the effect of both T_1 and T_2 relaxation times are minimized, or sequences susceptible to the flow of liquids [43].

If we consider the amplitude of the signal from two different points in two distinct tissues, it will decay at a different rate, producing a brighter or darker value in the final image. In addition to the common T_1 and T_2 weighted sequences, we also want to highlight the *fluid-attenuated inversion recovery* (FLAIR) sequence. Inversion recovery sequences allow us to suppress the signal of certain types of tissues [44]. In the BraTS dataset, a FLAIR sequence is used to suppress the signal of the cerebrospinal fluid. For other tissues, the contrast is similar to T_2 weighted images.

The signal that we record always corresponds to the (complex-valued) amplitude of some spatial frequency $k \in \mathbb{R}^3$ of the volume. To reconstruct an image, we need to measure a sufficient amount of different spatial frequencies to cover the frequency space, also known as *k-space*. We can, therefore, use the Fourier transform to relate the k-space with the image.

3.1.5 Artifacts

MRI-artifacts can have sources anywhere in the pipeline, be it the sequence, the spatial encoding, the anatomy inside and outside of the field of view, or even foreign objects like implants [45]. We will focus only on a few important types here. Figure 3.2 shows some examples of these. The most common image artifact comes from patient motion. In many cases, the k-space is acquired line by line, so a movement will result in an inconsistent k-space. It can be caused by gross patient motion but also breathing, blood

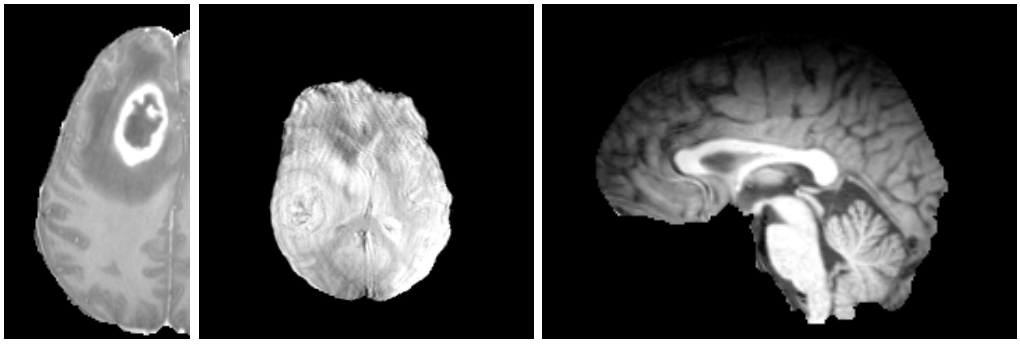


Figure 3.2: Examples of MRI artifacts visible in the BraTS dataset [25] , from left to right: (1) Gibbs artifact due to the sharp intensity changes along the inside of the skull, manifesting as wave patterns parallel to the skull surface. (2) Movement artifacts. Note that skull stripping was performed. Everything outside the brain mask is not visible. (3) Possibly a magnetic susceptibility artifact. The signal intensity drops towards the superior part.

flow, cardiac motion or peristalsis [39]. Furthermore, among the common artifacts are *Gibbs artifacts*, which manifest themselves as wave-like patterns along sharp edges in the image. Sharp edges have many high-frequency components, which may not be captured if the k-space is not sampled at sufficiently high spatial frequencies in the direction of the edge. This is usually observed in the direction of the phase encoding [40]. The final artifact type we want to mention is the *magnetic susceptibility artifact*: If a material or tissue has a higher magnetic susceptibility, that is, it becomes magnetized in the presence of a magnetic field, it can create a local inhomogeneity and lead to a loss or distortion of the signal [40, 39].

3.1.6 Clinical Applications

For certain applications, contrast agents are used. In MRI, they are usually Gadolinium-based. Gadolinium is a paramagnetic element [40]. It shortens the T_1-relaxation of nearby protons, which show up as hyperintense in T_1-weighted images. This is interesting to image blood vessels, but also for imaging brain tumors, especially when compared to the T_1-native scans: If the blood-brain barrier is disrupted, the blood, including the contrast agent can leak outside of the vessels and appear as hyperintense in that region [40].

3.2 Computed X-Ray Tomography

X-ray computed tomography (CT) is also commonly referred to just as *CT* uses X-rays to image the interior of the body. While with conventional radiography we can only record a projection through the object we investigate, CT uses multiple of these projections

from different angles to *computationally* infer what different parts inside look like, i.e., we perform *computed* tomography.

The main drawback of CT and X-ray is that X-ray radiation is *ionizing*, meaning it has enough energy to ionize the molecules, which damages biological tissue and increases the probability of cancer [46] and radiation induced hereditary diseases [47]. Therefore, for imaging, care should be taken to minimize the exposure while providing adequate image quality, and it should be considered whether the benefits of the exposure outweighs the risks associated with it. This is known as the ALARA principle ("as low as reasonable achievable") [48].

3.2.1 X-Rays

X-Rays are electromagnetic waves with a wavelength between roughly $10^{-11}\,\mathrm{m}$ and $10^{-8}\,\mathrm{m}$. Instead of wavelength λ or frequency ν, we usually prefer using the *energy* of a photon, in the context of radiology. These two quantities are directly related through the Planck-Einstein-relation

$$E = h\nu, \tag{3.5}$$

where $h \approx 4.136 \cdot 10^{-15}\,\mathrm{eV/Hz}$ is the *Planck-constant* [38]. In X-ray machines and CT-scanners, X-rays are produced using an arrangement of an anode and cathode with a resistively heated filament in a vacuum tube. In these tubes, electrons are accelerated with a high voltage in the range of $20\,\mathrm{kV}$ up to $140\,\mathrm{kV}$ and directed into an anode, typically made of tungsten or molybdenum, depending on the application [49]. The X-rays are then produced either as Bremsstrahlung when they lose kinetic energy by passing sufficiently close to a positively charged atomic nucleus or as characteristic radiation when they eject electrons from an inner shell of the anode atoms. In this case, another electron falls into the corresponding shell, giving off the lost potential energy as radiation. The characteristic radiation has very specific wavelengths, as they are a characteristic of the chemical element of the target. The Bremsstrahlung, however, covers a very wide band, as illustrated in Figure 3.3.

The attenuation of the X-ray beam is caused by different interaction types of the photons with matter. The most important one is the photo-electric effect, which is what we call the process in which a photon removes an electron from its shell and is absorbed in the process. Two further effects but with a smaller impact on attenuation, are the Compton scattering, where a photon interacts with an unbound or loosely bound (outer-shell) electron, and transfers some of the energy to the photon, and Raleigh scattering, where the photons do not have enough energy to ionize an atom and merely change direction [51]. All of these effects contribute to the reduction of the number of photons that arrive at the detector, i.e., the attenuation.

For the purposes of imaging, we do not model these interaction effects but rather use a simplified model: When an X-ray beam of intensity I_0 traverses an object, we assume that it is attenuated according to Beer-Lambert's law

$$\frac{I}{I_0} = \exp\left[-\int_0^1 \mu(\gamma(t))\gamma'(t)dt,\right] \tag{3.6}$$

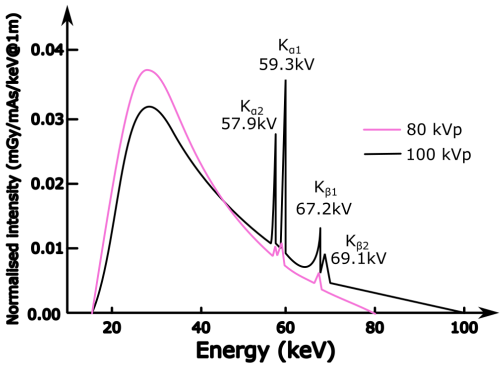


Figure 3.3: Emission spectrum of X-ray tube with tungsten anode with 24 mAs with a 3.04 mm aluminium filter. Courtesy of Raymond Chieng [50].

where μ is the linear attenuation coefficient of the material at point x and I is the intensity after the attenuation, and $\gamma : [0, 1] \to \mathbb{R}^3$ is a parametrization of the ray from X-ray source to the detector element [52].

Different types of tissues of the human body have different attenuation coefficients. This lets us use X-rays to image different parts of tissues of the human body. Traditionally, in X-ray, the attenuated radiation that exits the object being scanned is recorded using photographic film. Today, various types of electronic detectors are used. The most recent generation are *photon-counting* detectors, which can measure the energy of each photon individually, providing a higher resolution and lower noise while inflicting a lower dose to the patient [53].

The image we get from one such scan is a *projection*, as all the tissue one ray traverses contributes to its attenuation. Therefore, one point on the image represents the accumulation of all the tissue along a line.

3.2.2 CT-Scanners

Contrary to an X-ray radiograph, in a CT, we are not interested in a projection, but rather a *cross-sectional slice* through the body. While there are more advanced scanning methods, we will focus on this most fundamental type in this section for simplicity. In a typical clinical CT-scanner, the X-ray source and the detector are both mounted diametrically, and suspended on a rotating ring-shaped gantry, which is covered in a static fairing, as shown in Figure 3.4. This arrangement allows a patient table to be moved into the central opening to scan a certain cross-section of the body.

The detector, in its simplest form, is a single row of detector elements across from the X-ray source. If we were to record the detector's signal in one position, we would get a projection just like in X-ray radiographs, but just one pixel wide. To make an actual CT-scan, we must record many projections at slightly different angles, ideally covering

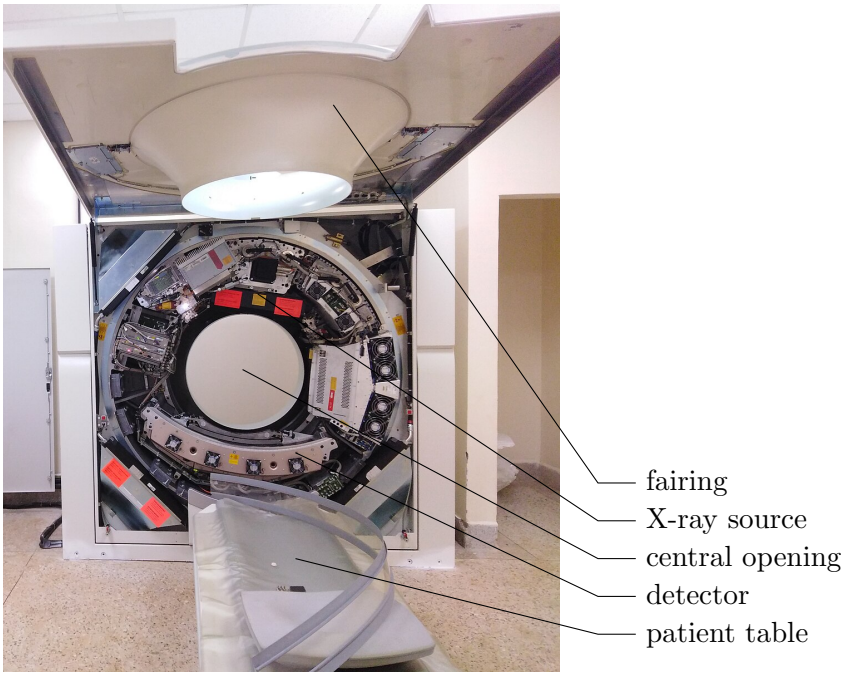


Figure 3.4: A CT-scanner with the fairing opened. Courtesy of user Obynel on Wikimedia Commons [54].

a whole rotation. If we group the rays by absolute angle, we get the so-called *sinogram*.

In Figure 3.5, we show a simulation of the projections along with the sinogram (i.e., parallel projections).

3.2.3 Image Reconstruction

The projections just contain the accumulated information along the rays. From these projections, we would like to reconstruct a cross-section. That is, we would like to find a corresponding attenuation coefficient for each location in the slice. Simple and computationally inexpensive reconstructions can be made using the *filtered backprojection*, based on the Fourier slice theorem. This method makes many simplifying assumptions and assumes highly idealized settings, that make it suffer from high noise levels and is especially susceptible to certain image artifacts (see Section 3.2.4) [55]. Modern algorithms are based on the reconstruction problem's algebraic formulation: If we discretize the image plane into voxels of absorption coefficients we'd like to determine, the integral in Equation (3.6) becomes a sum. Therefore, determining the value of the voxels amounts to solving a system of linear equations. Since the system is large and usually overdetermined and inconsistent, iterative numerical algorithms are used instead. These methods are based on accurately computing the forward projections, and their accuracy can be adjusted by varying the accuracy of the underlying modelling of the physics involved [55]. More recently, machine learning-based reconstructions have been

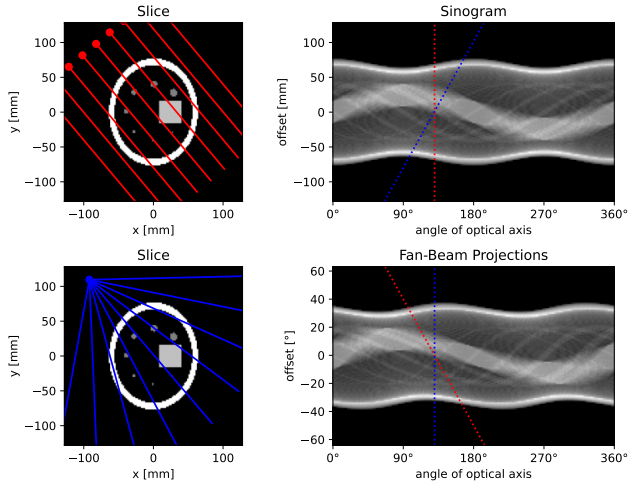


Figure 3.5: The simulated phantom slice, along with the sinogram in red and the fan-beam projections in blue. The location of the source for each beam is marked as a dot. The sinogram clearly shows that points in the slices correspond to sine waves in the sinogram space.

proposed [56].

The attenuation coefficients μ (see Equation (3.6)) we get through the reconstruction are normalized into the dimensionless *Hounsfield unit* (HU) μ^* with

$$\mu^* = 1000 \cdot \frac{\mu - \mu_w}{\mu_w} \tag{3.7}$$

where μ_w is the absorption coefficient of water. This is a handy convention to identify tissue types.

For image reconstruction, we usually make the simplifying assumption that the absorption coefficient is not dependent on the photon energy (or, alternatively, that the beam is monochromatic), as with most scanners, it is not possible to quantify the energy of the detected photons [55]. This leads to artifacts like the *beam hardening artifacts* (see Section 3.2.4), but also means that the HU scale is dependent on many factors like the actual spectrum of the X-ray beam and the detector properties [57]. Still, it roughly allows us to tell apart or identify certain types of tissues.

3.2.4 Artifacts

CT scans are susceptible to various types of artifacts [38]: A common artifact is patient motion. Because the different projections are recorded step by step over time, we find discontinuities across the reconstructed slices. Furthermore, there are artifacts caused by physics: Compton scattering causes photons to change directions and be recorded by a detector element that was not in line with the initial trajectory. This can, to some

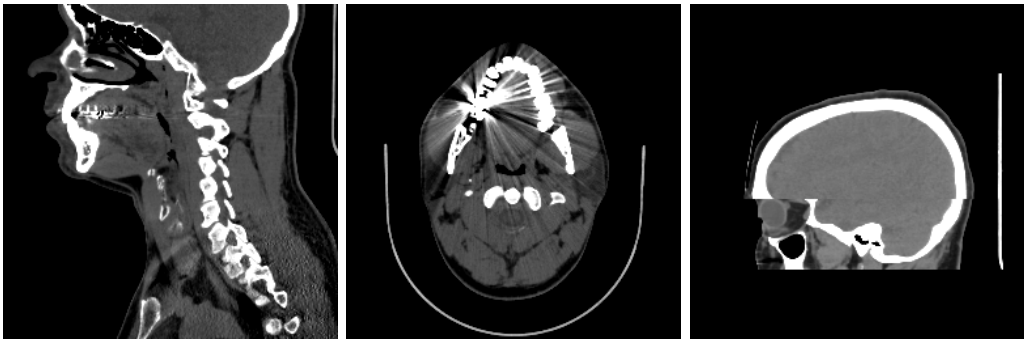


Figure 3.6: Examples of CT-artifacts visible in the CQ500 dataset [33] , from left to right: (1) photon starvation artifact at the height of the shoulders, as the X-rays have to traverse more tissue. At the height of the teeth we see metal artifacts due to dental implants. (2) metal artifacts due to dental implants (3) movement artifact, visible as discontinuity across the entire slice

degree, be mitigated by placing a collimator over the detector, which aims to absorb scattered photons, i.e., photons that do not come from the direction of the X-ray source.

The X-ray beams contain a very wide spectrum of radiation. Beam hardening artifacts occur because not every part of this spectrum gets absorbed the same. Especially dense objects absorb low-energy ("soft") radiation disproportionally more, resulting in streaking and cupping artifacts in the reconstructed image. This is visible, for instance, in metal implants. A way to reduce this effect is to place filters in front of the X-ray source that filter out low-energy photons. Another common type of artifact is photon starvation. If a very absorbent material is scanned, e.g., lead with a very high absorption coefficient, it can lead to the detector element behind to detect next to no signal. This can also lead to streaks in the image [58]. We show some examples of artifacts in the data we used in Figure 3.6.

3.2.5 Clinical Applications

CT is a modality that is very well suited for the use in emergency medicine, as the scans are very fast (depending on the type of scan, taking mere seconds [55]). It has a diverse range of applications. For instance, it can be used for cancer screening and staging, for detecting traumatic injuries such as hematoma, hemorrhages and fractures, for detecting pulmonary conditions and also for angiography [59]. The main advantages over MRI are the high resolution, short scan times and safety regarding metal objects.

While the radiation dose for CT is a lot higher than for an X-ray, CT scans are still considered to have low-level radiation exposure, i.e., far below 100 mSv per scan [60]. While exposure events above this threshold are considered harmful, it is more difficult to quantify harm in low-level exposures, as it depends, among other things, on the affected organs and the age and gender of the patient. But a general rule of thumb says that the risk of developing fatal cancer increases by 0.5% per 1000 mSv of dose [60].

Chapter 4

Deep Learning – A Short Introduction

In this chapter, we will first motivate the use of machine learning, which also encompasses deep learning. From there, we will introduce the basics of deep learning, which will serve as the foundation for all models developed in this thesis.

4.1 Machine Learning and Deep Learning

When we want to use a computer to automate certain tasks, we have to tell the computer exactly what to do. We have a recipe – an *algorithm* – in mind that we want the computer to execute, and we write programs to communicate the algorithm to the computer. For many tasks, it is easy to develop algorithms, like performing arithmetic. Other tasks are seemingly simple for humans, like telling apart images of cats and dogs, reading handwritten text, or detecting cancerous lesions in MRI, however, it is not straightforward to come up with an algorithm that allows a computer to perform these tasks. This is the point where machine learning shines. Tasks that we humans can do, but are difficult, if not impossible, for us to explicitly formulate as algorithms, are tasks that we solve by relying on our experience or intuition. Machine learning enables us to do something similar but with computers: Instead of formulating an algorithm, we define a space of possible algorithms to perform our task. Using a large amount of data, we search or optimize for a good algorithm within our space of possible algorithms. In many cases, we actually just consider deterministic algorithms, i.e., a space of functions.

To illustrate this, let us consider a well-known example of the CIFAR10 dataset [61] comprising natural color images with a resolution of 32×32 pixels of ten different classes, among which are cats and dogs. For humans, it is straightforward to tell apart cats and dogs with high accuracy, but it is not straightforward to explicitly instruct a computer on how to perform this task. One possible machine learning method would be performing

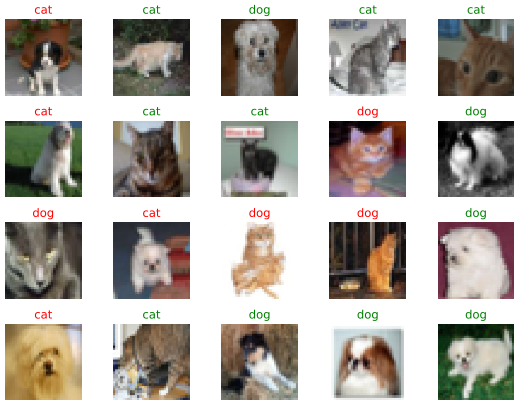


Figure 4.1: Some samples of the CIFAR10 test set with the prediction of the linear regression ("cat" or "dog"). The prediction is colored green if it is correct and red if false, according to the labels provided in the test set. The linear regression model with a decision threshold of 0.5 achieved an accuracy of 0.594.

a linear regression on the labels.[1] We show some examples of the test set along with the predictions of linear regression in Figure 4.1.

Let us use the convention of labelling cat = 1 and dog = 0. The linear regression provides us with an optimal affine function

$$f_w(x) = w_0 + \sum_{i=1}^{32\times32\times3} w_i x_i \tag{4.1}$$

mapping the input x – in our case, a vector containing the color intensities of an image – to the predicted output $f_w(x)$ - here a scalar value.

Optimal in the context of linear regression usually means that it minimizes the squared difference between the function evaluated on a given image x, and the ground truth label y of said image

$$\mathcal{L}(w) = (y - f_w(x))^2. \tag{4.2}$$

We can now use this function f, to classify images. We predict that an image contains a cat if the predicted value is below 0.5, and a dog if it is above. Hence, instead of telling the computer explicitly how to perform this classification, we first defined an *objective*, which consisted of minimizing the sum of squared errors of the predictions. Then we also defined a space of possible algorithms. In our case, the algorithm consists of evaluating a linear function followed by a case distinction via thresholding (i.e., applying the Heaviside function). So, in short, the space of algorithms was equivalent to the space of linear functions mapping the color intensities to a scalar. Finally, we searched for a satisfactory algorithm in that space, by trying to minimize our objective on our dataset.

[1] Whether it is an appropriate method for the task is up to debate, as it is certainly not the best performing. Here, we use it just as an illustrative example.

For the special case of linear regression, we are actually in luck, in that we can find an optimal solution with respect to the objective, but as we will see later, this is practically unattainable for many search spaces.

When working with neural networks and applying them in deep learning, we do also take this approach. We design a search space by defining some "architecture" of a function that contains some amount of free parameters. In the example in Equation (4.1), the free parameters are given by $\{w_i\}_i$. However, these architectures can be a lot more complex. A core problem when working with neural networks is finding a satisfactory combination of architecture and objective, that is suited for the task at hand, along with a corresponding optimization strategy.

4.2 Neural Networks

In machine learning, the study of neural networks has undergone a long development. We can go back as early as the mid twentieth century: Initially, neural networks were an attempt to mathematically model biological neurons [62, 63]. Generally, artificial neurons are modeled as nodes that aggregate multiple activations of other neurons as an input and produces an output based on that. This simulates the behavior of biological neurons, which aggregate the signals received via the dendrites, and if that aggregated signal is strong enough, it will be sent through the axon to the synaptic terminals, where other neurons are connected and can pick up the signal. In Figure 4.2, we show a graph representation of a neuron of an artificial neural network along with a simplified representation of a biological neuron by which the former was inspired.

The simplest (artificial) neural network, therefore, consists of a single neuron. In typical networks, each signal is represented by a real number. The aggregation is done by multiplying each incoming signal x_i with a weight w_i, and summing those products up. The output y is then modeled by subtracting a threshold value t (sometimes also referred to as *bias*), and then passing the result through an activation function φ [62].

$$y = \varphi\Big(\sum_i w_i x_i - t\Big) \tag{4.3}$$

Classically [62, 63], the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ has been used as a smooth approximation of the binary Heaviside function h, which models the firing/non-firing state, as shown in Figure 4.3. Later, the rectified linear unit (ReLU), and variations thereof, became popular, as it improved the convergence and is cheaper to compute [64].

To make more complex networks, first, multiple neurons can be used in parallel to build a *wider* layer. Then, multiple of those layers can be chained in series to actually make a *deep* network, i.e., a network of more than one layer of neurons. This neural architecture is called a *multilayer perceptron* (MLP) [65]. Deeper network enabled the modeling of more complex relationships. For instance, with one layer of neurons, solving the XOR-problem is impossible. The XOR-problem consists of modeling the exclusive-or

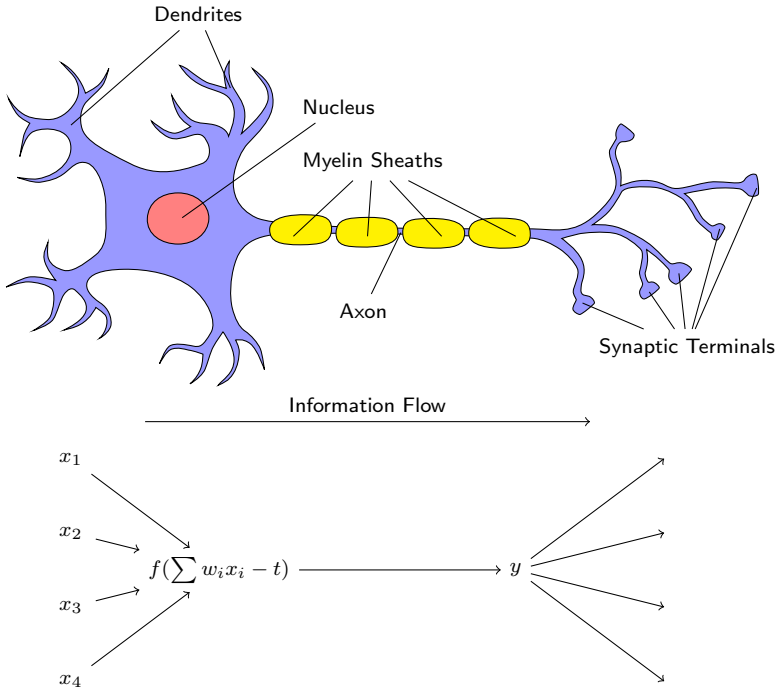


Figure 4.2: The upper illustration shows a simplified schematic representation of a biological neuron of the cerebral cortex, the lower graphic shows how such a neuron was modeled in artificial neural networks. The assumption is that the dendrites receive the information from other cells, the information gets weighted and accumulated, and if that signal is strong enough, the neuron is "firing", i.e., transmitting the information to other cells via the synaptic terminals.

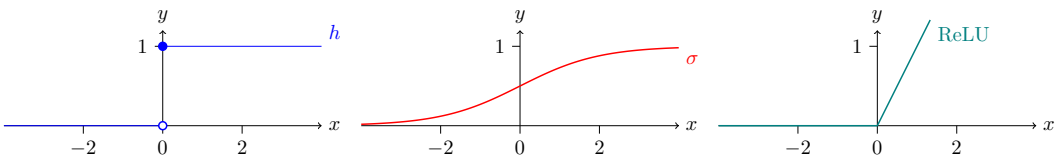


Figure 4.3: Plots of three typical activation functions $h(x) = \mathbf{1}_{x \geqslant 0}(x)$, $\sigma(x) = 1/(1+e^{-x})$, and $\mathrm{ReLU}(x) = \max(0, x)$.

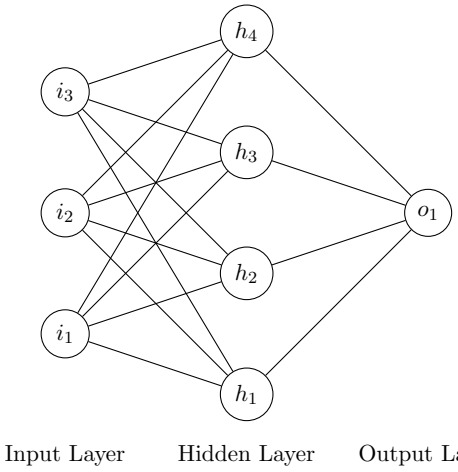


Figure 4.4: A prototypical neural network with input- and output layers as well as one hidden layer. It can be considered a two- or three-layer network, depending on whether we omit or include the input layer in the count. For this work, we use the former convention.

function,

$$\begin{aligned} \mathrm{XOR}(0,0) &= 0 \quad \mathrm{XOR}(0,1) = 1 \\ \mathrm{XOR}(1,0) &= 1 \quad \mathrm{XOR}(1,1) = 0 \end{aligned} \tag{4.4}$$

which can be understood as a classification problem with a *nonlinear* decision boundary. With a single layer network, f let us say that we decide that a sample $x \in \mathbb{R}^2$ belongs to class 1 if $f(x) \geqslant c$ for some constant c. Then the decision boundary is the boundary ∂P of the set of positive decisions

$$P = \{x \mid f(x) \geqslant c\}. \tag{4.5}$$

The boundary ∂P is linear (i.e., represented by a line $ax_1 + bx_2 + c = 0$), if a strictly monotonic activation function is used, like, e.g., the mentioned sigmoid function σ.

When we use multiple neurons in parallel, i.e., in the same layer, we can simplify the notation using matrices and vectors. Now each row $W_{i,:}$ of the matrix $W \in \mathbb{R}^{n \times m}$ represents the weights of one neuron, and the corresponding entry b_i of the vector $b \in \mathbb{R}^n$ refers to the bias. By convention, we flip the sign of the bias compared to Formula Equation (4.3), and the activation function $\varphi : \mathbb{R} \to \mathbb{R}$ is applied element wise.

$$y = \varphi(Wx + b) \tag{4.6}$$

Where $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. This formulation of an MLP is the basis of many types of neural networks.

4.2.1 Convolutions

The development of MLPs also lead to the idea of using other linear transformations. Linear filters are a cornerstone of digital signal processing, including image processing, and are a natural candidate for an alternative linear transformation. This is especially interesting if the input x is, e.g., a time series or an image instead of a vector of arbitrary features. In the context of neural networks, this idea surfaced first in the Neocognitron [66].

Since then, the types of architectures kept evolving, but many deep neural networks keep mostly the same building blocks: On the one hand, linear (or more accurately *affine*-) functions with trainable weights, and fixed nonlinear functions. Examples of the latter include, for instance, the various activation functions, the z-normalization in Batch-Normalization and the related normalization blocks [67, 68, 69, 70], maximum pooling operations [71], softmax blocks to convert an arbitrary vector into a probability distribution, attention blocks [72] in sequence processing, dictionary lookups for vector embeddings [73] and many more.

4.2.2 Deep Networks

In their early form, a network with more than one layer of neurons, i.e., more than one hidden layer, was considered *deep* [65]. As the previously mentioned XOR-problem illustrates, the depth of a network allows for learning more complex relationships or features, as was demonstrated most prominently with the ResNet [74]. The ResNet, which is a deep convolutional network, demonstrated that an increased depth allows for – at the time – unprecedented performance increase in image classification competitions.

Deep networks are nontrivial to optimize, as they exhibit a highly non-convex loss landscape [65]. What enabled ResNets to be so deep was their use of the previously introduced Batch Normalization [67] and their proposed *residual* architecture.

Compared to a conventional network $f = f_N \circ \ldots \circ f_2 \circ f_1$ they proposed using additive skip connections known as *residual connections*, that is, building their network $g = g_N \circ \ldots \circ g_2 \circ g_1$ with

$$g_i(x) := f_i(x) + x \tag{4.7}$$

In the same year, the now ubiquitous U-Net [75] architecture was proposed for image segmentation tasks. As opposed to a ResNet, it consists of an encoder-decoder architecture. It also featured skip connections but used concatenation instead of addition. These were originally proposed to aid in the preservation of the detailed structures throughout the network. They have, however, also a similar effect on the training as the skip connections in ResNets.

4.2.3 Objective Functions and Losses

The *training* of a neural network f_ϑ of some given architecture and the given data $\mathcal{D} = \{d_1, \ldots, d_m\}$ refers to the optimization of the network parameters ϑ with respect

to some loss function $\mathcal{L}$. These loss functions are usually chosen depending on the task at hand, and when formulated as *Loss* they are implicitly assumed to be minimized. For instance for a regression problem with $d_i = (x_i, y_i)$, i.e., predicting continuous scalar values y_i from some input x_i, frequently the *mean squared error* (MSE) between the prediction $f_\vartheta(x_i)$ and the desired label y_i is used:

$$\mathcal{L}(\mathcal{D}, \vartheta) = \frac{1}{m} \sum_{k=1}^{m} \|f_\vartheta(x_i) - y_i\|_2^2 \tag{4.8}$$

Another typical task is classification: The goal is to predict the membership of some input x_i to a finite set of different classes $\mathcal{C} = \{c_1, \ldots, c_k\}$. To this end, the cross-entropy between two probability distributions is used for training: We design our network to output k different scalar values $\hat{y} = f_\vartheta(x) \in \mathbb{R}^k$ for some given input x. Using the so-called *softmax*-function, we can transform an arbitrary vector into a discrete probability distribution. It consists of applying exponential functions to map all entries to positive values and then dividing them by their sum to make those values sum to 1, i.e., provide a valid probability distribution:

$$q := \text{softmax}(\hat{y}) = \frac{1}{\sum_{i=1}^{k} e^{\hat{y}_i}} (e^{\hat{y}_1}, e^{\hat{y}_2}, \ldots, e^{\hat{y}_k}) \tag{4.9}$$

The label y_i of the given data x_i is encoded using the *one hot-encoding*

$$p := 1_{\mathcal{C}}(y_i) = (0, \ldots, 0, 1, 0, \ldots, 0) = (\delta_{c,y_i})_{c \in \mathcal{C}} \tag{4.10}$$

The *cross-entropy* loss is defined as the cross entropy H between these two probability distributions

$$\mathcal{L}(\mathcal{D}, \vartheta) = H(p, q) = -E_p[q] = -\sum_{i=1}^{k} p_i \log(q_i) \tag{4.11}$$

These losses presented in Equations 4.8 and 4.11 are two of the most fundamental ones and are the basis for many more complex losses or combinations thereof.

4.2.4 Optimization

We will now discuss how these objective functions are optimized, for this, let us consider the case of the MSE in Equation (4.8). The goal is minimizing $\mathcal{L}$ with respect to ϑ, that is, by choosing the architecture of f_ϑ, we define the space of functions, or equivalently, the space of parameters that we optimize in.

While any kind of optimization strategies could be applied, the by far most popular methods are *gradient*-based optimizers, i.e., methods involving the derivative with respect to the parameter, or, in this case, *gradient descent*. This, however, requires that the architecture is differentiable (at least with respect to the parameters ϑ).[2]

[2] As we saw in Figure 4.3, the ReLU activation is not differentiable at $x = 0$. In practice, just defined the derivative at that point as 0 or 1 which are subgradients at $x = 0$ [76]. There are other non-differentiable operations like vector quantization [73] that use different mechanics to perform a useful optimization [76].

The most simple form of the gradient descent algorithm is defined as follows:

$$\vartheta^{(i+1)} := \vartheta^{(i)} - \lambda \nabla_\vartheta \mathcal{L}(\mathcal{D}, \vartheta^{(i)}) \tag{4.12}$$

In this expression, $\vartheta^{(i)}$ is the point in the optimization space of the current iteration, and λ is the so-called *learning rate*.

The losses that we want to minimize are posing another challenge as they are computed across the whole dataset $\mathcal{D}$. As $\mathcal{D}$ often contains hundreds to millions of data points, it is not practical to compute the loss value and the corresponding derivatives in every optimization step. Instead, *stochastic* gradient descent (SGD) is used. This means that in the computation of the loss, only a random subset $\mathcal{D}^{(i)} \subset \mathcal{D}$ of the training data $\mathcal{D}$ is used in Equation (4.12). The cardinality $|\mathcal{D}^{(i)}|$ is often referred to as *batch size*.

In comparison to SGD, more advanced optimizers have been developed, like the popular *Adam* [77], that try to estimate the first or also second moment of the stochastic gradient using moving averages to accelerate the convergence.

With gradient descent methods, we do need to choose a suitable starting point $\vartheta^{(0)}$. Some heuristics have been proposed on how to choose that starting point on a layer-by-layer basis. They are based on the idea that we want the layer f_ϑ to preserve some characteristics of a distribution of inputs and outputs [78, 79]. The same idea can also be applied to more complex blocks that consist of multiple layers.

4.2.5 Computation of Gradient

The computation of the gradient $\nabla_\vartheta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \vartheta}$ at some given parameter set ϑ can be performed in various ways. An efficient algorithm for this is the *backpropagation*-algorithm. While it has been developed multiple times, sometimes only for specific types of networks [80], there have been debates over the inventor [81]. In the modern deep learning frameworks, we now use the general notion of the more descriptive *backward-mode automatic differentiation* [81]. Because the details of the mathematical derivation are rarely explicitly explained (especially for general compute graphs, not just trees), this section is an attempt to explain it a little more rigorously based on the chain rule.

The quantity we want to minimize, i.e., the loss $\mathcal{L}$, can be written as a composition of functions

$$\mathcal{L} = (l_n \circ \ldots \circ l_2 \circ l_1)(\vartheta) \tag{4.13}$$

with $l_i : \mathbb{R}^{a_i} \to \mathbb{R}^{b_i}$, since the weights ϑ are leaf-nodes of the computation graph of $\mathcal{L}$. Here $a_i, b_i \in \mathbb{N}$ for $i = 1, \ldots, n$ are the number of input- and output dimensions. Note that, depending on the architecture, the l_i do not necessarily correspond to the layers of a network: They refer to the functions that are defined by the intermediate graphs between subsequent *graph-cuts*, i.e., the intermediate stages in a topological sorting of the computation graph, as is visualized in Figure 4.5. In this diagram, the l_i would correspond to the functions

$$\vartheta = \underbrace{\begin{bmatrix}\vartheta_a\\ \vartheta_b\\ \vartheta_c\\ \vartheta_d\end{bmatrix}}_{=\lambda_0} \overset{l_1}{\mapsto} \underbrace{\begin{bmatrix}a\\ \vartheta_b\\ \vartheta_c\\ \vartheta_d\end{bmatrix}}_{=\lambda_1} \overset{l_2}{\mapsto} \underbrace{\begin{bmatrix}a\\ b\\ \vartheta_c\\ \vartheta_d\end{bmatrix}}_{=\lambda_2} \overset{l_3}{\mapsto} \underbrace{\begin{bmatrix}b\\ c\\ \vartheta_d\end{bmatrix}}_{=\lambda_3} \overset{l_4}{\mapsto} \underbrace{[d]}_{=\lambda_4} \overset{l_5}{\mapsto} \underbrace{[e]}_{=\lambda_5} = \mathcal{L}, \tag{4.14}$$

where we used the shorthands $a = a(x, \vartheta_a)$, $b = b(a, \vartheta_b)$, $c = c(a, b, \vartheta_c)$, $d = d(b, c, \vartheta_d)$ and $e = e(y, d)$.

To formalize this, consider the compute graph $G = (N, E)$ where N are the nodes. The edges between nodes correspond to the values that are passed between the nodes, and the nodes represent the functions that process these values. Incoming edges represent the arguments, outgoing edges represent their (possibly vector-valued) values. Let us use the function v to refer to the values of an edge. For instance, $v((l_0, l_1)) = \vartheta$ With this formulation, we implicitly include two artificial nodes, namely a source node T emitting the weights ϑ and a sink node L, which takes the variable of which we want to compute the gradient with respect to the input. As the inputs to the network are considered constant, we absorb them as constants directly into the nodes to which they contribute to.

Since G represents a computation, we know it is a directed acyclic graph. We can, therefore, compute a topological order, i.e., we can enumerate the nodes as $T = l_0, l_1, l_2, \ldots, l_n, L = l_{n+1}$ such that we only have edges from nodes of lower numbers to nodes of higher numbers, i.e., for all $(l_i, l_j) \in E$ we have $i < j$. We now consider the cut $C_k = (L_k, H_k)$ after node l_k by setting $L_k = \{l_i \in E \mid i \leqslant k\}$ and $H_k = \{l_j \in E \mid k < j\}$. Due to the topological ordering, E contains only edges from nodes in L_k to nodes in H_k. we collect them as

$$E_k = \{(l_i, l_j) \in E \mid i \leqslant k \text{ and } k < j\} \tag{4.15}$$

and concatenate $\lambda^{(k)} = (v(e))_{e \in E_k}$, i.e., we understand $\lambda^{(k)}$ to be in $\mathbb{R}^{d_k}$ for some $d_k in \mathbb{N}$.

Hence, the gradient with respect to ϑ can be computed recursively using the chain rule. To illustrate this, we write the intermediate values as $\lambda^{(i)} = (l_i \circ l_{i-1} \circ \ldots \circ l_1)(\vartheta)$. In general, the chain rule is written as

$$J_{f \circ g}(x) = J_f(g(x)) J_g(x) \tag{4.16}$$

with the Jacobian matrix J_f defined for $f : \mathbb{R}^n \to \mathbb{R}^m$ as

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \cdots & \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix}. \tag{4.17}$$

Therefore, for $j < i$, we can write

$$J_{l_i \circ \ldots \circ l_j}(\lambda^{(j)}) = J_{l_i}(\lambda^{i-1}) \cdot J_{l_{i-1}}(\lambda^{(i-2)}) \cdot \ldots \cdot J_{l_{j+1}}(\lambda^{(j)}) \cdot J_{l_j}(\lambda^{(j-1)}) \tag{4.18}$$

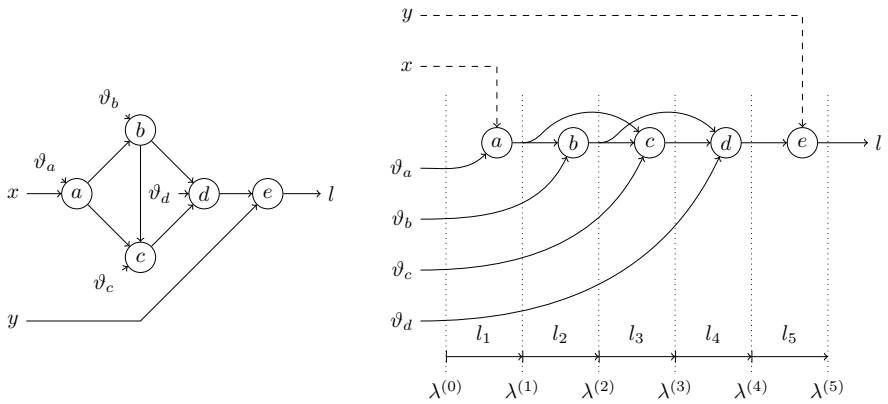


Figure 4.5: On the left, we display a simple compute graph of some network, e.g., a loss function. The circled nodes a,b,c,d represent functions and the arrows represent the values. The values x,y can be thought of as data, the values ϑ_a,ϑ_b,ϑ_c,ϑ_d,l as weights or parameters of the function. For the purpose of optimization, we consider the value l that we'd like to minimize as a function of the weights ϑ, but not of x and y. If we pull these as inputs to the left, and sort nodes topologically, i.e., in an order they can be evaluated in, we can decompose the whole network into a chain of function compositions l_i, as also shown on the right and in Equation (4.14). This decomposition can be done by considering graph cuts, shown as dotted lines between the subsequent nodes in the topological ordering.

After we evaluated $l = \mathcal{L}(\vartheta)$ (i.e., the *forward pass*), we store all the intermediate activations $\lambda^{(i)}$. The backpropagation algorithm makes use of this during the backward pass: With equation Equation (4.18), it is easy to see that in every step i we compute

$$\nabla_{\lambda^{(i)}}\mathcal{L} = J_{l_n}(\lambda^{(n)}) \cdot \ldots \cdot J_{l_{i+1}}(\lambda^{(i)}). \tag{4.19}$$

We can step by step compute $\nabla_{\lambda^{(n)}}\mathcal{L}$, $\nabla_{\lambda^{(n-1)}}\mathcal{L}$,..., $\nabla_{\lambda^{(0)}}\mathcal{L} = \nabla_{\vartheta}\mathcal{L}$, by traversing the composition of the l_i backwards, and computing recursively

$$\nabla_{\lambda^{(i-1)}}\mathcal{L} := \nabla_{\lambda^{(i)}}\mathcal{L} \cdot J_{l_i}(\lambda^{(i-1)}). \tag{4.20}$$

This is the heart of the backpropagation algorithm.

4.2.6 Graphics Processing Units

In 2007 NVIDIA released CUDA [82], which is an application programming interface (API) for performing general purpose computations on their graphics processing units (GPUs). In the 2010s, the use of GPUs for performing the computations in neural networks became widespread and enabled speeding up the training and inference processes. In 2014 the cuDNN library was released, which provided CUDA implementations of basic building blocks of neural networks [83]. Shortly after, the two most popular deep learning frameworks TensorFlow (in 2016) [84] and PyTorch (in 2017) [85] have been published, which simplified the usage of GPUs for applications involving neural networks.

4.2.6.1 Matrix Multiplication

While GPUs have been more limited in the types of instructions they can perform than central processing units (CPUs), they are designed to perform a large number of computations in parallel. In graphics processing, we frequently have to perform the same operation many times, parallelly and independently, for instance, when we compute vertex or pixel shaders. These shaders are functions that compute certain properties, e.g., the color, of a geometric object (i.e., a vertex of a mesh) or a pixel. A common operation in this context is the matrix-vector multiplication, specifically with 4×4 matrices, as many geometric transformations in 3D can be expressed as such using homogeneous coordinates [86].

This poses a great opportunity for neural networks, as discrete (finite dimensional) linear transformations can be expressed as matrix-vector multiplications. The simplest example is the general matrix-vector multiplication in MLPs. As convolutions are linear transformations, they can also be represented as matrix multiplications. Due to the special structure of the corresponding matrices, there are multiple different algorithms perform the corresponding matrix-vector multiplication. They include the naive matrix multiplication, the convolution via fast Fourier transforms and the Winograd algorithm [87]. The choice is dependent on the hardware, but also on the size of the input and convolution kernels [87]. However, to illustrate how neural networks can profit form these parallelized matrix multiplications, we will stick to the first example of general matrix-vector multiplications.

Matrix-vector multiplications of larger sizes can be broken into smaller ones, and therefore be efficiently computed on GPUs. To illustrate, consider a matrix in $A \in \mathbb{R}^{2n \times 2n}$ with a vector $x \in \mathbb{R}^{2n}$ can be decomposed in submatrices $A_{ij} \in \mathbb{R}^{n \times n}$ and $x_i \in \mathbb{R}^n$ with $i, j \in \{1, 2\}$ as

$$Ax = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right] = \left[\begin{array}{c} A_{11}x_1 + A_{12}x_2 \\ \hline A_{21}x_1 + A_{22}x_2 \end{array}\right] \tag{4.21}$$

To put this into context, we can consider the NVIDIA A100 GPU as an example [88]: It has 108 streaming multiprocessors (SMs) with each 64 FP32 CUDA Cores. When we perform a matrix multiplication

$$F^{N \times K} \times F^{K \times M} \to F^{N \times M}, \tag{4.22}$$

in this device (with cuBLAS 11), it is most efficient if the matrix dimensions N, K, M are multiples of 2^7 if the entries F are 8-bit integers, 2^6 if the entries are 16-bit floating point numbers, 2^5 for 32-bit tensor-float numbers, and 2^4 for IEEE 64-bit floating point numbers. Hence, it is most efficient if we can partition the matrices we want to multiply into submatrices what have these sizes. Furthermore, in this particular device, it is most efficient if the number of submatrices is (or is slightly smaller than) a multiple of 108.

4.2.6.2 Memory

The system memory (random access memory (RAM)) is – along with the register file – the main memory used by the CPU [89]. When the CPU needs to process data that is stored on a disk (e.g., hard disk drive (HDD) or solid-state drive (SSD)), it will first get loaded into RAM, from which the CPU can access and modify it.

GPUs have a more fine-grained subdivision of different memory types and parts, but here, we want to give just a high-level overview based on the NVIDIA Ampere architecture [89]. While many GPUs have similar components, the actual architecture depends on the specific GPU. If we want to process data on the GPU (assuming a discrete GPU that is independent of the CPU), we first have to transfer it to the main graphics memory (video-RAM (VRAM)). Reading data from the disk into RAM is a relatively slow process, usually limited by the disk reading speed. Transferring data from RAM to VRAM is already a lot faster. The A100 has a bandwidth of up to 200 Gbit/s. Accessing the VRAM is even faster, up to 1555 GB/s for the 40 GB model [90]. The SMs can access the VRAM directly, but they also have each their own transparent cache, called the level 1 (or L1) cache. Furthermore, the VRAM access is also being cached using the level 2 (or L2) cache. On the NVIDIA A100 the VRAM has a size 40 GB or 80 GB - depending on the model, an L2 size of about 40 MB and an L2 size of 192 KB [90].

It is therefore evident that the closer to the SMs the data that we want to process can be stored, the faster the processing can be performed. To train a neural network using a GPU, we typically store the model weights on the VRAM, and our training data on a disk. If the dataset is small enough, it might even be possible to load it all into RAM. Otherwise, during an optimization step, the data is read into RAM, possibly using

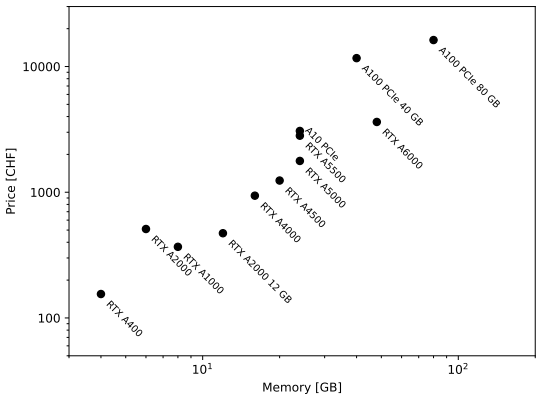


Figure 4.6: Prices and Memory of discrete NVIDIA GPUs with Ampère architecture. We can estimate a relationship of Price $\propto$ Memory$^{1.5}$. (Prices according to digitec.ch as of 2024-08-29)

multiple threads in parallel [91]. From there, the data can be transferred to VRAM, to be processed by the SMs. Ideally, we are therefore able to store all arrays of the compute graph in VRAM, but also the the network parameters. Depending on the type of optimizers, we might even need to store multiple values per parameter [77]. Especially for medical image processing, where we frequently use 3D volumes, sometimes with multiple channels, sometimes even time series, the memory management becomes an important consideration. As we have shown in [92] (see Chapter 7), it is not straightforward to adapt deep learning models, that have been proposed for 2D data to 3D, if we take the memory constraints into account. There are general approaches like *gradient checkpointing* [93], that can trade memory consumption against re-computations, and, therefore, time and energy. This is a technique that stores only some intermediate activations in the compute graph and recomputes them again during the backward pass. However, they do waste resources in recomputing the same intermediate activations a second time, which is why we are interested in techniques that are inherently more memory efficient.

In modern GPUs, the amount of memory they are equipped with, mostly correlates with their other performance parameters, and therefore is also linked to their price. In Figure 4.6, we compared the amount of memory and price of some popular recent NVIDIA GPUs. We can conclude that for enabling the training of large deep learning models, requiring substantial amount of computational resources, the hardware can be a major contributor to the costs.

Chapter 5

Applications of Neural Networks

We have discussed the components of neural networks and their training in the previous chapter. In this chapter, we dive deeper into their applications, and how they are used to build more complex models.

5.1 Tasks and Supervision

In the previous chapter, we already mentioned a few well-known tasks that can be solved using neural networks. We have seen that the optimization of these networks is often done using gradient descent, which is performed to minimize some loss function in a simple supervised learning setting.

Many supervised machine-learning problems can be split into classification and regression tasks. Given some input, regression tasks boil down to predicting one or more continuous values. In contrast, in a classification task, we try to predict a discrete value, i.e., one of multiple possibilities. Examples of the classification task include handwritten character recognition [94] or object classification in images [61]. Similarly, image segmentation tasks can also be viewed as classification tasks, considering that we perform a pixel-wise classification, i.e., we classify pixels as belonging to a certain segment. Furthermore, the prediction of an entire image can be seen as a pixel-wise regression task.

There are, however, other tasks in which the desired output given some input is not available for the data we train on. Conversely, there are also supervised tasks where the labels of the training data are not directly used to train the network.

While there are many supervision settings, we will now attempt to classify some of those that are relevant for image processing: Broadly speaking, we can distinguish methods that make predictions on an image level (e.g., classification), bounding-box level (e.g., object detection) or pixel level (e.g., segmentation). Furthermore, with respect to the supervision, we can distinguish methods that use pixel-level labels, bounding boxes, image-level labels, or no labels at all. We talk about *supervised* or *fully supervised* training, we can distinguish whether the type of predictions match the type of labels we use during training. It is worth noting, however, that for instance in the context

		training labels			
		none	image level	bounding box	pixel level
predictions	image level	U	F	F	F
	bounding box	U	W	F	F
	pixel level	U	W	W	F

Figure 5.1: An attempt of categorizing different training and prediction regimes into three supervision types: unsupervised (U), weakly supervised (W), and fully supervised (F).

of anomaly detection (see Section 5.4), the term *unsupervised* is used when a model is trained on *normal* data only, i.e., when an image level label is known, but only one class is being used. Similarly, *weak* supervision is sometimes equated with *semi*-supervison [95]. However, in this, we will use the term *weak* to describe the setting, where we have less information available for training than we will predict for inference, e.g., when we use image level labels to make pixel-level predictions. This usage is more common in the context of medical images, as we will see in Section 5.4. We use the term *semi*-supervised for methods that train on both labeled and unlabeled data.

Table 5.1 shows an attempt to summarize the types of training labels and predictions we mentioned above and classify them into the different supervision regimes we discussed.

5.2 Generative Models

Generative models are models that have some way of *generating* new data similar to the data used in training. In many cases, this takes the form of a mapping between a well-known distribution, like a standard normal distribution, and a distribution that is given by just some realizations, i.e., the training data.

5.2.1 Variational Autoencoders

Autoencoders are networks trained using an auto-encoding objective, i.e., they are trained to output whatever is given as an input. While that might sound trivial at first, the key idea is that they have a *bottleneck* in their architecture. Usually, autoencoders are a composition $d \circ e : \mathbb{R}^N \to \mathbb{R}^N$ of an encoder network $e : \mathbb{R}^N \to \mathbb{R}^M$ and a decoder network $d : \mathbb{R}^M \to \mathbb{R}^N$ with $M \ll N$. The *bottleneck* refers to the reduced dimensionality M between the two networks, which prevents an identity mapping of the whole input domain $\mathbb{R}^N$. If both d and e are linear, M imposes an upper bound on the *rank* of $d \circ e$, i.e., a linear dimensionality reduction. If we choose d and e as deep neural networks, then $d \circ e$ represents a nonlinear dimensionality reduction. In any case, we try to compress the information in the high-dimensional input- and output-space $\mathbb{R}^N$ into a lower-dimensional latent space in $\mathbb{R}^M$.

So in general, given some data point $x \in \mathbb{R}^N$ the *reconstruction* objective of an

autoencoder is usually formulated as

$$\mathcal{L}_{\text{rec}} = d(x, (d \circ e)(x)). \tag{5.1}$$

where d can be any distance measure but frequently is chosen as the l^2 distance

$$d(x, y) = \|x - y\|_2^2 \tag{5.2}$$

Variational autoencoders (VAEs) work similarly, but they are endowed with a special bottleneck [96]. Instead of passing the output of the encoder directly to the decoder, we use the encoder to estimate the parameters of some probability distribution. We then pass a realization of this distribution to the decoder. Usually, a normal distribution with diagonal covariance is used. Therefore, the encoder is predicting a mean vector and the diagonal covariance matrix, i.e., $\mathbb{R}^{2M}$ values. The key point now is that in addition to our *reconstruction* objective $\mathcal{L}_{\text{rec}}$, we also add an objective that encourages the output of the encoder e to follow a known distribution, i.e., a standard normal distribution. This is done using a bound to the Kullback-Leibler divergence [96].

As we encourage the latent space to follow a known distribution, we can generate new samples after the training just by passing new realizations of the known distribution into the decoder. A notable application of this principle is the Vector-Quantized-VAE [73], that introduced some more refined techniques that enabled the generation of high-resolution images.

5.2.2 Generative Adversarial Networks

Similarly to the decoder in VAEs, Generative Adversarial Networks consist of a generator network $g : \mathbb{R}^M \to \mathbb{R}^N$ mapping a known distribution (again, e.g., a standard normal distribution) into the space of the training set [97]. It was first demonstrated on small images (up to 32×32 pixels), but can also be applied to other kinds of data. During the training, the second network $d : \mathbb{R}^N \to [0, 1]$, called a *discriminator*, is trained to distinguish between the images generated by the generator g ("fake") and the images from the dataset ("real"). The generator network g, however, is trained in an *adversarial* way, that is, to produce output that makes the discriminator d classify them as images originating from the dataset, even though they are fake.

While the principle is simple, it has major flaws: Even when the generator successfully fools the discriminator, we are in no way guaranteed to get any meaningful generated outputs, as the discriminator can be sensitive to features that are not what humans perceive as important. Also, there is no incentive for the generator to produce diverse output. Another major difficulty is the instability of the training of GANs due to the adversarial training objective [98]. Various techniques have been proposed, to alleviate these issues issue [99, 100, 98].

5.2.3 Autoregressive Models

When generating samples in autoregressive models, the model will – as the name implies – consider the already generated content to produce an additional part. For instance,

a text model could generate the next letter based on the previous ones or an image model could generate a pixel based on previous ones. A notable example of the latter is the PixelCNN [101]. It is an autoregressive image model that employs a recurrent neural network that predicts new pixels based on some hidden state and the previously generated pixels. Another notable class of autoregressive models comprises transformer-based models [72]. They have been used extensively in large language models (LLMs) like BERT [102], GPT [103] and llama [104]. While pure transformer architectures have been used in image processing tasks [105, 106, 107], in generative image models, they are currently used more frequently as auxiliary components in convolutional architectures [108, 109, 110].

5.2.4 Implicit Neural Representation

Images or shapes can mathematically be modeled implicitly as a scalar- or vector field $I : \Omega \to C$, mapping a coordinate in a 2D- or 3D space $\Omega \subset \mathcal{R}^d$, to e.g., a color in $C = \{0, 1, \ldots, 255\}^3$, or occupancy value in $C = \{0, 1\}$. Typically, we discretize images as pixels or voxels, and an image is therefore represented as an array of values, which can be interpreted as image with $\Omega \subset \mathbb{N}^d \to C$. Implicit neural representations (INRs), however, model this mapping directly using a neural network

$$f_\vartheta : \Omega \to C \tag{5.3}$$

mapping the input coordinates $x \in \Omega$ to some output value $y = f_\vartheta(x) \in C$. The same approach can also be used for representing audio with $\Omega \subset \mathbb{R}$, or video with $\Omega \subset \mathbb{R}^{d+1}$, containing an additional time dimension [111]. In the simplest case of representing a single image with an INR, we can train the network f_ϑ by minimizing the reconstruction error, using, e.g., an ℓ^2 distance

$$\mathcal{L} = \sum_{x \in \Omega} \| f_\vartheta(x) - I(x) \|_2^2 \tag{5.4}$$

5.2.4.1 Applications

INRs have an advantage in that they can be directly used with any resolution or even irregularly sampled data. A salient example is the use of an INR to represent 3D scenes as proposed with the NeRF method [112]: The input to the network consists again of the coordinates of the point we want to query, but also of the viewing direction. The output consists of a color and a transparency term. A volume rendering pipeline is being used to render perspectives of the whole 3D scene.

The same approach can be adapted to CT reconstruction [113, 114]. The view dependence, as well as the color output, can be omitted, and the volume rendering is replaced with a physical absorption model of the X-rays, e.g., the Beer-Lambert law as in Equation (3.6). During training, we can, therefore, render the projections we would get from the current state of the INR network, and compute a loss between the rendered and the real projections.

The representation has been used in a wide variety of applications, like image compression [115], image inpainting [116], novel view synthesis [112, 117], 3D shape reconstruction [118, 119], CT reconstruction [114, 120, 113], semantic segmentation [121], shape generation [122], shape completion [123], and surface reconstruction from point clouds [124].

As already alluded to above, INRs have been shown to perform well for irregularly sampled data, as they are not dependent on a grid as for instance pixel images. In medical imaging, especially in tomographic methods like OCT, CT or MRI, it is common to have different in-plane and out-of-plane resolutions. INRs have successfully been used for super-resolution tasks using two anisotropic MR scans with different contrasts and different plane orientations [125].

Since each point $x \in \Omega$ can usually be processed independently, the network f_ϑ can be as simple as an multilayer perceptron (MLP), and does not require more complex building blocks like convolutional layers. It has, however, been demonstrated that INRs with traditional activation functions are slow to converge and cannot provide very fine detail [111], i.e., they have a *spectral bias* [126]. Two approaches have been investigated to solve this issue. The first uses a *positional encoding* [112]: The network f_ϑ was outfitted with an additional function that maps the coordinates to the *Fourier features*

$$\gamma(p) = (\cos(2^0\pi p), \sin(2^0\pi p), \cos(2^1\pi p), \sin(2^1\pi p), \ldots, \cos(2^{L-1}\pi p), \sin(2^{L-1}\pi p)). \quad (5.5)$$

where $p \in [-1, 1]$ is one coordinate and $L \in \mathbb{N}$ the number of encoding levels. This function is first applied to each input coordinate independently before being passed to the parametrized layers. Alternative positional encodings include Fourier features with random frequencies [127], or multi-resolution hash encoding [128] have been proposed.

The second approach focuses on the activation functions. Periodic functions, like a sine in SIREN [111], have been shown to outperform traditional ones like the sigmoid or the ReLU. A modification thereof has been proposed [113], which instead uses parametrized wavelets of the form

$$\sigma(x) = \exp(i\omega_0 x) \cdot \exp(-|s_0 x|^2). \quad (5.6)$$

where auxiliary layers predict ω_0 and s_0. Further modifications include variable frequency activations [129], or implicitly via intermediate second-order polynomials [130].

5.2.4.2 Latent Space

In some applications, the architecture of INRs acts as a prior for images, shapes or scenes. In these settings we optimize a separate set of parameters for each image or volume [112, 114, 120, 125, 111]. In other applications, it can be interesting to train an INR equipped with some latent space in the form of an additional input. In this case the the trained network acts as a more specific prior for a whole distribution of images [121, 118, 122].

It has been shown that we can use denoising diffusion models [131, 132] or generative adversarial networks [133] to directly generate the parameters of an INR. This can be

understood as falling under the umbrella term of *hypernetworks*, that is, networks that predict the weights of other networks (the INRs in this case). The subspace of the parameter space of the INRs that these models cover, can be considered as a latent space [111].

A latent space $\Lambda \subset \mathbb{R}^l$ can also be modeled more explicitly, by designing the INR network to accept further input $z \in \Lambda$, that encodes some information about the image it represents as

$$f_\vartheta : \Omega \times \Lambda \to C. \tag{5.7}$$

A separate z can then be used for each training set image and – depending on the application – jointly optimized with the network. For the subsequent application of the INR, we also need to be able to project new images into this latent space Λ. It is possible to do so by jointly training an encoder network, mapping an input image to the latent space Λ in an auto-encoding objective [134].

Another approach that does not rely on an additional encoder is the direct optimization of the latent representation, that is, given some image $I : \Omega \to C$, we optimize

$$\hat{z} = \arg\min_{z \in \Lambda} \sum_{x \in \Omega} d[f_\vartheta(x, z), I(x)], \tag{5.8}$$

using some distance function d. In the medical domain, this method has also successfully been applied for shape modelling and reconstruction, as well as semantic segmentation [118, 121]. As the evaluating network only performs the *decoding*, this type of INRs is also called *auto-decoder*.

Therefore, when equipped with a meaningful latent space, INRs are not only a representation of data that can be generated by an external mechanism – such diffusion models or GANs that we mentioned above – but can also be used for various applications, including generative tasks [111, 119, 134] if we introduce a meaningful latent space Λ. In Chapter 8, we discuss the application of INRs to model the brain development process of neonates over time.

5.2.5 Denoising Diffusion Models

Denoising Diffusion models have been proposed as generative models to generate high-quality images [109]. In their most basic form, the model is trained to predict a real-looking image given a sample of Gaussian noise [109], that is, it is trained similarly to a denoising autoencoder. The major difference is that to generate a single sample, we must apply the same network multiple times (typically up to $T = 1000$). While we start with Gaussian noise, the output of the network is again fed back into the network in each step, producing a more and more denoised image. While Gaussian noise was initially proposed and is therefore highly popular, we can also formulate this process with other probability distributions [135, 136, 137], or completely different methods of image degradation [138].

5.2.5.1 General Principle

To illustrate the principle, we first consider the *diffusion* process (a.k.a. the forward- or noising-process) q as proposed in [109]. Using their notation[1], we define the conditional distribution

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{5.9}$$

where I is the identity matrix. The sequence of β_t can be chosen in various ways, provided that $\beta_1 = 0$ and $\beta_T > 0$ [109]. In practice, we chose $0 \leqslant \beta_T \ll 1$, $\beta_T \gg \beta 1$, and β_t is a monotonic function of t, e.g., linear, interpolating between β_1 and β_T. This means we diffuse x_{t-1} to generate x_t using

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon \tag{5.10}$$

where ε is a realization of $\mathcal{N}(0, I)$. Since ε is random, there is a whole distribution of possible x_t. It is possible to reformulate the process with the condition on x_0 instead of x_{t-1} with $\alpha_t := 1 - \beta_t$ and $\overline{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ as

$$q(x_t \mid x_0) := \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I). \tag{5.11}$$

This equation shows us that our noising process q does indeed "interpolate" between the standard normal distribution $\mathcal{N}(0, I)$ and the image x_0, or in other words, it degrades the image x_0 by adding Gaussian noise. If we consider x_t, x_{t-1}, and ε as random variables, we can use Equation (5.11) to deduce that

$$\frac{x_t - \sqrt{\overline{\alpha}_t}x_0}{\sqrt{1-\overline{\alpha}_t}} \overset{d}{=} \varepsilon \sim \mathcal{N}(0, I) \tag{5.12}$$

This relation is at the core of how we employ the network. We train the network $\varepsilon_{\vartheta,t}$ to predict the (z-normalized) noise ε we have to remove it from x_t to get x_0. However, to get to x_{t-1} we only make a small step in that direction.

Let us first look at the training objective and then at the *denoising* process $p_{\vartheta,t}(x_{t-1}|x_t)$. Given a random variable following the training set distribution $x_0 \sim \mathcal{D}$, and a random variable $\varepsilon \sim \mathcal{N}(0, I)$, and some random diffusion step $t \in \{1, ..., T\}$, we aim to minimize

$$\mathbb{E}\Big[\|\varepsilon - \varepsilon_{\vartheta,t}(\underbrace{\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon}_{\overset{d}{=}x_t})\|^2\Big]. \tag{5.13}$$

Note that the argument of $\varepsilon_{\vartheta,t}$ actually is the distribution of x_t. Hence, to perform the training, we sample an image x_0 from the training set, a realization ε of $\mathcal{N}(0, I)$, and compute $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$. The key in this process is now that for this particular

[1] The rigorous distinction between random variables and their realization is not made in favor of a shorter notation. Equation (5.9) can be read as follows: The variable x_t is a realization of the random variable X_t which has distribution $X_t \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, which is a function of x_{t-1}. We will, however, use the notation "$X \overset{d}{=} Y$" to indicate that both random variables X and Y have the same distribution.

x_t, we *know* the realization of the noise ε, which we can use as a supervision. We therefore aim to minimize

$$\mathcal{L} = \left\| \varepsilon - \varepsilon_{\vartheta,t}(\sqrt{\overline{\alpha}_t} x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon) \right\|^2 \tag{5.14}$$

with respect to ϑ. To generate a new image from $x_T \sim \mathcal{N}(0, I)$, in [109], the denoising process is given by

$$x_{t-1} \overset{d}{=} \sqrt{\overline{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\overline{\alpha}_t}\varepsilon_{\vartheta,t}(x_t)}{\sqrt{\overline{\alpha}_t}} \right) + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2}\varepsilon_{\vartheta,t}(x_t) + \sigma_t \varepsilon_t \tag{5.15}$$

where $\varepsilon_t \sim \mathcal{N}(0, I)$ and

$$\sigma_t = \sqrt{\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}} \cdot \sqrt{1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}}} \tag{5.16}$$

with $\overline{\alpha}_0 = 1$.

As explained in [110], this process is stochastic, as in every time step t we add ε_t. But if setting $\sigma_t = 0$ for all t, as proposed in [110], this process becomes deterministic and only depends on x_T. Then Equation (5.15) simplifies to the DDIM sampling scheme [110] given by

$$x_{t-1} = \sqrt{\overline{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\overline{\alpha}_t}\varepsilon_{\vartheta,t}(x_t)}{\sqrt{\overline{\alpha}_t}} \right) + \sqrt{1-\alpha_{t-1}}\varepsilon_{\vartheta,t}(x_t). \tag{5.17}$$

5.2.5.2 Conditioning

In many applications, we are not just interested in the unconditional image generation, but we want to steer the network to an output with some desired property.

Let us consider the case of semantic segmentation: An unconditional diffusion model could be trained to generate arbitrary plausible segmentation masks of the objects we want to segment. However, this alone is not yet helpful if we want to segment objects in a given image. A simple way – that has been proven effective for this task – consist of concatenating [139, 140] of the condition to the input. We concatenate the image we would like to segment to the input x_t of the denoising network $\varepsilon_{\vartheta,t}$ in every step t, as shown in Figure 5.2.

Many other conditioning methods have been proposed and can be divided into three categories: First, we can condition the image generation process in the latent space of the denoising network [141, 142]. Second, we can use the gradient of an auxiliary network to guide the image generation towards a desired property [143, 144]. Finally, the deterministic nature of the DDIM [110] sampling can also be used to encode a given image in the noise, which can be used as a conditioning mechanism [145, 146].

5.3 Segmentation

The image segmentation task can be understood as a per-pixel classification task. Given an image $I : \Omega \to C$ and a set of classes $\Gamma = \{c_0, \ldots, c_n\}$, we would assign a class to

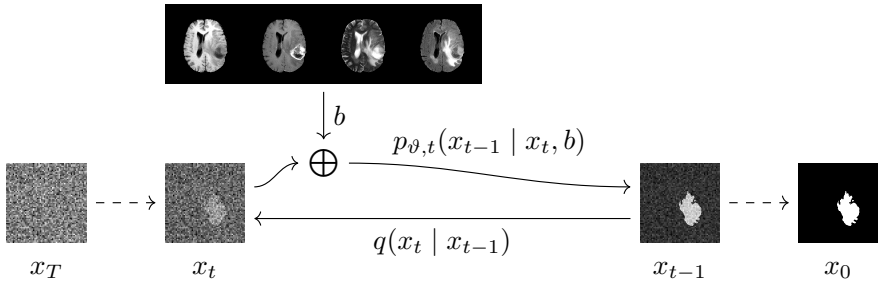


Figure 5.2: The conditioning of a denoising process to generate brain tumor segmentations [92] (see also Chapter 7).

each pixel $x \in \Omega$, that is, we would like to find a function $S : \Omega \to \Gamma$. In the example of the *brain tumor segmentation challenge* (BraTS) [25], the images I consist of four MR sequences, i.e., $C = [0, \infty)^4$ and Γ consists of four classes as depicted in Figure 2.6 (three classes of tumor tissue along with the background). To perform this task with a neural network, we typically use a network that outputs an image of the same dimensions as the input. We usually encode the classes using a *one-hot* encoding, that is, the network has to be designed to output the same number of channels as we have classes (including background).

5.3.1 Segmentation Models

The most widespread network architecture for segmentation is the U-Net [75] introduced in 2015. The architecture and its variations have become a fundamental building block for many tools, including denoising diffusion models [109, 110, 147]. For segmentation tasks, the nnU-Net was proposed [147], which is *self-configuring*, i.e., it chooses many parameters based on the data and task at hand, and has been shown to outperform many manually tuned U-Nets [148].

While the U-Net architecture is common, there are also many other approaches. It has been shown that we can use INRs to segment anatomical structures in 4D (3D+t) data [121]. Segmentation models have also been proposed using recurrent networks [149] and GANs [150]. A denoising diffusion model has been proposed for brain tumor segmentation [140]. Due to the inherently stochastic nature of denoising diffusion models, we can perform the segmentation of a given input image multiple times and use different seeds for the random generators to get slightly different segmentation masks. In that sense, we can think of segmentation as a *generative* task. It has been shown, that the fusion of these segmentations improves the performance in terms of various metrics but also allows capturing the uncertainty within the model [140].

The same effect has been shown in humans: When an expert is asked to segment the same image multiple times, the segmentations will differ, which is known as the *intra-rater variability*. The differences between different raters is called the *inter-rater variability*. However, if multiple segmentations are fused, the performance can be im-

proved [151].

Chapter 7 discusses the segmentation approach proposed in [140] and what further techniques were necessary to implement it in a memory-efficient way for 3D volumes, as opposed to 2D slices.

5.3.2 Losses and Metrics

An important question is what loss functions we use to train these supervised segmentation models, and what validation metrics we use to compare their performance.

In the following part, we will discuss some loss functions and metrics, for which we want to introduce some notation: For simplicity, we assume that we have two classes: one class c_1 for the feature we are interested in, and c_0 for the background. Furthermore, we denote the ground truth and the prediction as $Y, \hat{Y} : \Omega \to S$ respectively. As Y and $\hat{Y}$ each partition Ω into foreground and background, we can alternatively consider the foreground set as the preimage of Y and $\hat{Y}$. We denote them with $A = Y^{-1}(c_1) \subset \Omega$ and $B = \hat{Y}^{-1}(c_1) \subset \Omega$. Furthermore, for a given set $X \subset \Omega$, we use the notation $X^c \subset \Omega$ for the complement, i.e., $X^c = \{x \in \Omega \mid x \notin X\}$.

The (pixel-wise) cross entropy (see Section 4.2.3) can be used as a loss function for segmentation tasks, but many other loss functions and combinations thereof have been proposed [152, 153]. We can broadly categorize them into three different categories [152]: pixel-level, region-level and boundary-level. The pixel-level losses include the MSE and the cross entropy, as they can be computed for each pixel independently.

The region-level losses include the *Accuracy*

$$ACC = \frac{|(A \cap B) \cup (A \cup B)^c|}{|\Omega|}, \tag{5.18}$$

borrowed from classification tasks, measuring the relative amount of pixels that has accurately been classified. A disadvantage is that for small classes (i.e., $|A| \ll |\Omega|$), the accuracy can be high even if the foreground class has not been captured ($A \cap B = \emptyset$).

A popular metric that ameliorates this issue is the Dice score (also known as F1-score)

$$DSC = \frac{|A \cap B|}{\frac{1}{2}(|A| + |B|)}, \tag{5.19}$$

and equivalently, the Intersection-over-Union

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \tag{5.20}$$

They normalize the intersection area, such that we get a *relative* error. They in turn have the disadvantage of producing low scores for small foreground classes, even if only a small area has been misclassified. To be used as a loss $\mathcal{L}$ in practice, both of them have to be formulated as a differentiable function of the prediction, and need to be inverted,

e.g., using $\mathcal{L}_{DSC} = 1 - DSC$. If we assume $\hat{Y} : \Omega \to [0, 1]$ and $c_0 = 0, c_1 = 1$ (i.e., $Y : \Omega \to \{c_0, c_1\} = \{0, 1\}$), we can define

$$\mathcal{L}_{DSC} = 1 - \frac{2 \sum_{x \in \Omega} Y(x) \hat{Y}(x) + \varepsilon}{\sum_{x \in \Omega} Y(x)^2 + \sum_{x \in \Omega} \hat{Y}(x)^2 + \varepsilon} \tag{5.21}$$

where $\varepsilon > 0$ is a constant for avoiding numerical issues when a class is empty, i.e., when $Y(x) = 0$ for all $x \in \Omega$.

Finally, the boundary-level metrics include the Hausdorff distance

$$\begin{aligned} HD &= \sup \left[\{ d(a, B) \mid a \in A \} \cup \{ d(b, A) \mid b \in B \} \right], \\ \text{where } d(S, r) &= \inf_{s \in \partial S} \|r - s\|_2 \end{aligned} \tag{5.22}$$

is the set-point distance. It is possible to also formulate it in a way that is differentiable with respect to $\hat{Y}$ [154], in order to be able to use as a loss in gradient-based optimization. As a validation metric, it is also common to report the HD95, i.e., the "95th-percentile Hausdorff distance". This is a variation of Equation (5.22), where the supremum is replaced by the 95th percentile, as the pure Hausdorff distance is susceptible to noise and single pixel outliers. From these examples, we can conclude that there is a lot of freedom in the choice of loss functions for segmentation tasks. Often it is worth using a combination of multiple loss functions [152].

For the subsequent validation, the measures highly depend on the task at hand and on the clinical question. Furthermore, relationships within the data have to be taken into account, for instance if for some patients multiple images are included. The *Metrics Reloaded* consortium, analyzed many of these potential issues and provides an extensive framework to guide through the choice of metrics [153].

5.4 Anomaly Detection

Anomaly detection is the task of detecting *anything* that is not considered *normal*. There is, however, no clear path on how we can go about that, and it also depends on the type of data we can use for training and of the type of predictions we want to make. In this section, we focus on anomaly detection that also includes anomaly localization within an image.

We focus on methods that try to extract more specific information about the *location* of an anomaly in an image. This is an information that is not available as a label during training, that is, we consider the weakly- and unsupervised regime. As we try to extract additional information, the prediction of labels (if they are available) is insufficient. To train a model, we instead have to rely on a surrogate task.

As a first example, we can consider [155], which proposed an unsupervised approach. An autoencoder is trained to reconstruct healthy ("normal") brain MRIs. To localize the anomalies, the residual, i.e., the difference between the input and the reconstruction, is considered. This principle is called *restoration*, and an important concept that many anomaly detection methods are based on. Similar methods were proposed by [156,

157, 158], using a variational autoencoders instead. [159] compared multiple of these autoencoder-based restoration approaches using common architectures.

Another class of approaches includes GAN-based weakly supervised anomaly detection. [160] used CycleGAN [161] as a basis for anomaly detection and localization. The method proposed a translation between images with and without brain tumors and implicitly included a masking mechanism to provide the tumor localization. We consider this a weakly supervised method, as the presence of brain tumors has been used as an image-level label.

DeScarGAN [162] proposed another weakly supervised GAN-based approach, which however used the restoration directly for the localization, as did the autoencoder-based approaches mentioned above. f-AnoGAN [163] also uses a GAN but in an unsupervised setting. The restoration is again used for anomaly localization, but also includes latent-space restoration for image level anomaly detection.

Denoising Diffusion models have also been leveraged for anomaly detection. [164] uses a weakly supervised approach for anomaly localization by leveraging a classifier network to perform a gradient-based conditioning. The image undergoes a partial noising and denoising process using the DDIM-scheme [110]. The gradient of the classifier is used to condition the denoising process to generate a healthy-looking image. Similarly, AnoDDPM [145] uses the same partial noising and denoising scheme, but in an unsupervised manner: The diffusion model is only trained on normal images. Therefore, the denoised images also appear normal and are then also used for a restoration-based localization.

We can conclude that there is a variety of methods that finally use a restoration as a surrogate task for anomaly detection. [165] criticizes the use of restoration-based anomaly detection. They mention that those methods frequently were evaluated on datasets using brain-MRIs that include FLAIR sequences, in which brain tumor or multiple sclerosis (MS) lesions can appear hyperintense. The authors showed, that a simple approach consisting of skull stripping, histogram normalization, thresholding and filtering connected components by size can outperform many restoration-based anomaly detection methods.

The "medical-out-of-distribution"-challenge [166], the participants are asked to come up with general *out-of-distribution*-detection, which is not necessarily specific to a certain type of anomaly. While the test data is not public, the challenge is known to include lesions, but also, technical or artificial anomalies, to pose a more general anomaly detection problem. [167] and [168] tackled this problem by training classic segmentation models in a supervised manner, but also adding in synthetic image corruptions, for which possible segmentations are available, hence we can identify the self-supervision on synthetic corruptions as a surrogate task. In [169] (see Chapter 6), we proposed a patch-based approach to tackle this problem. The surrogate task we proposed is identifying the location of a small patch within the image. When this is performed for as many patches as desired, we can measure how far the predictions were off, and use this to detect and localize anomalies.

Chapter 6

Position Regression for Unsupervised Anomaly Detection

In this publication, we propose a surrogate task for unsupervised anomaly detection and localization. The method is evaluated on a dataset of co-registered head CT scans that were collected from patients with suspected brain injuries [33]. The proposed network is trained on CT of patients that did not exhibit any injuries. It is trained using a surrogate task which consists of extracting small patches of the image, and predicting their position within the image, i.e., *position-regression*. We show that the proposed network fails to recover the position of patches that exhibit fractures and hemorrhages, and can solve the task well on patches of tissue without any injuries. This enables the detecting and localizing anomalies. The proposed architecture can use relatively small networks, and with the patch-wise processing, can be run with very little memory.

Publication. The following paper was presented at the *Medical Imaging with Deep Learning* (MIDL) Conference, July 2022, Zürich. It was published as part of the conference proceedings [169].

Proceedings of Machine Learning Research 172:1–13, 2022 Full Paper – MIDL 2022

Position Regression for Unsupervised Anomaly Detection

Florentin Bieder[1] FLORENTIN.BIEDER@UNIBAS.CH
Julia Wolleb[1] JULIA.WOLLEB@UNIBAS.CH
Robin Sandkühler[1] ROBIN.SANDKUEHLER@UNIBAS.CH
Philippe C. Cattin[1] PHILIPPE.CATTIN@UNIBAS.CH
[1] *Department of Biomedical Engineering, University of Basel*

Abstract

In recent years, anomaly detection has become an essential field in medical image analysis. Most current anomaly detection methods for medical images are based on image reconstruction. In this work, we propose a novel anomaly detection approach based on coordinate regression. Our method estimates the position of patches within a volume, and is trained only on data of healthy subjects. During inference, we can detect and localize anomalies by considering the error of the position estimate of a given patch. We apply our method to 3D CT volumes and evaluate it on patients with intracranial haemorrhages and cranial fractures. The results show that our method performs well in detecting these anomalies. Furthermore, we show that our method requires less memory than comparable approaches that involve image reconstruction. This is highly relevant for processing large 3D volumes, for instance, CT or MRI scans.

Keywords: medical image analysis, anomaly detection, unsupervised

1. Introduction

In recent years, anomaly detection has become an essential direction of research in medical image analysis. Compared to supervised segmentation methods, anomaly detection methods do not rely on pixel-wise annotations but on image-level labels instead. This leads to a much simpler way of annotating the training data and reduces the human bias to the model.

We can distinguish two major types of anomaly detection methods in the literature: The first type only uses data that is considered normal. In terms of medical images, these are images of healthy subjects. The second type additionally requires examples of anomalous data (Battikh and Lenskiy, 2021; Wolleb et al., 2020), and can also be considered as weakly supervised methods. However, in this work we will focus on the first type that uses only normal data for the training.

The most widely used methods for image anomaly detection are based on reconstruction errors (Baur et al., 2021). These methods aim to capture the distribution of the training set of healthy subjects by learning a low dimensional representation of the input and the reconstruction from this representation back to the original image. The core idea is that the correct reconstruction of the input will fail in some regions if an anomaly is present in the input image. Comparing the output with the input will provide a reconstruction error in the pixel space, which can be used as an indicator for anomalies (Chen and Konukoglu, 2018; M.D. et al., 2018; Baur et al., 2021). A challenge in these methods is generating an output image of high quality and rich in detail. This requirement contributes to the

computational cost in terms of training time and hardware requirements, i.e., GPUs and memory. (Tong et al., 2021) have also observed that autoencoders can have a bias towards data that can easily be reconstructed and are sensitive to outliers in the training set.

This paper presents a novel anomaly detection method for medical images based on position regression. In contrast to the image reconstruction-based methods our approach operates on patches of the input image. Instead of learning a reconstruction of the patch in pixel space, our method predicts the location of the input patch in the original image. Our method is trained only on data of healthy subjects and implicitly learns the distribution of the data as a result of the position regression task. During inference, a significant error in the position prediction of a patch indicates that an anomaly is present within that patch, which was not present in the training distribution. A detailed overview of our method is shown in Figure 1. We evaluate our method on the head CT dataset presented by (Chilamkurthy et al., 2018), which contains images of patients with intracranial haemorrhages and cranial fractures.

Coordinate regression problems have been explored for point-of-interest localization (Nibali et al., 2018) or to propose bounding boxes in object detection (Girshick et al., 2014; Ren et al., 2015). However, these methods focus on finding the location of particular objects within the input image. Compared to this, our method predicts the coordinates of the input patch with respect to the remaining part of the source image. (Lei et al., 2021) proposed a method of estimating positions of patches to locate specific organs or other anatomical structures within whole-body CT scans. In contrast to our method, they simultaneously feed two patches into their network and perform a regression over the relative position of the two patches.

2. Method

The method we propose is based on position regression. From an input volume image $I \in \mathbb{R}^{N \times N \times N}$ we extract patches $p_{\mathbf{x}} \in \mathbb{R}^{s_p \times s_p \times s_p}$, with a patch size of s_p at position $\mathbf{x} \in \mathbb{R}^3$ within I. The voxels in I can be indexed using three dimensional coordinates (x, y, z) with the coordinates $x, y, z \in \{0, 1, 2, \ldots, N-1\}$. For our purposes we normalize these coordinates to the range $[0, 1]$ such that $\mathbf{x} = (x, y, z) \in [0, 1]^3$.

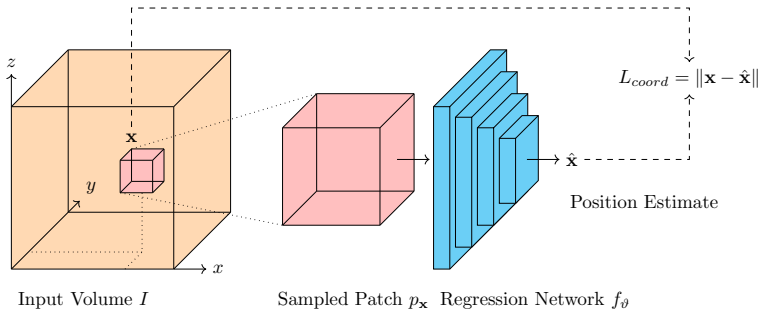


Figure 1: Conceptual overview of the our proposed method.

Using Cartesian coordinates works well for head CT scans, as they always have the same structures in a similar position. For scans of other anatomical structures different that may have different poses, other coordinate systems might be better suited, for instance, a barycentric coordinate system based on some key points.

Given some patch, our network $f_\vartheta : \mathbb{R}^{s_p \times s_p \times s_p} \to \mathbb{R}^3$ with parameters ϑ is trained to output an estimate $\hat{\mathbf{x}} = f_\vartheta(p_\mathbf{x})$ of the source position $\mathbf{x}$ of the given patch, as shown in Figure 1. Thus we can consider the training of the network as solving a regression problem.

For training, we iterate over our training set: In each iteration, we consider an input volume I. We randomly sample coordinates $\mathbf{x}$ from the input volume I and extract the surrounding patch $p_\mathbf{x}$. Then we pass the patch through the network to get the estimated coordinates $\hat{\mathbf{x}} = f_\vartheta(p_\mathbf{x})$ and use the ℓ^2-norm of the difference between the coordinates of the patch $\mathbf{x}$ and the estimated coordinates $\hat{\mathbf{x}}$ as our training loss

$$L_{coords}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|.$$

In practice, we use multiple patches in each iteration and average the individual losses L_{coords} to get our training loss that we optimize. These patches are sampled independently from a uniform distribution over input coordinates.

During inference, we compute an output volume E of the same size as the input I. For every voxel $I_\mathbf{x}$ at coordinates $\mathbf{x}$ we sample the surrounding patch $p_\mathbf{x}$ of the input I, and pass it through the network to get an estimate $\hat{\mathbf{x}}$. Note that the patches centered at the coordinates of two neighbouring voxels will overlap. Then we define the output volume E by computing the reconstruction error for each voxel as $E_\mathbf{x} = L_{coords}(\mathbf{x}, \hat{\mathbf{x}})$.

This allows us to see the output volume E as an error map. Each $E_\mathbf{x}$ shows how well the network f_ϑ could predict the position of the patch $p_\mathbf{x}$ centered at $\mathbf{x}$. If the value of a voxel in the output volume E is above a certain threshold, the network failed to predict the correct position of the associated input patch. This is the case if the input patch exhibits a structure that is not present at that position in the training images. Therefore, we can check for high values in the output volume E to find regions that appear anomalous.

2.1. Architecture

The network we used for the patch position regression (PPR) has a generic image classification network architecture. The detailed structure is displayed in Figure 2. The architecture is parametrized by a network size parameter m to consider a whole range of networks with a different number of parameters. This parameter determines the number of channels in the convolutional blocks as well as the size of the affine layers. We define two blocks, "Downsample" and "Residual" that are used throughout the network. Here "AvgPool" stands for average pooling with a kernel of size 2 and stride of size 2 in every direction. "Conv(c, k, s)" stands for a 3D convolution with c output channels, a kernel size of k and a stride of s with spectral normalization. "Linear(n)" is an affine transformation with a codomain of dimension n.

3. Experiments

All runs were trained for 2000 epochs with the Adam optimizer (Kingma and Ba, 2014) with a learning rate $\mathrm{lr} = 10^{-4}$. We selected the learning rates empirically based on the training

#	Block
1	Residual(m)
2	Downsample(m)
3	Residual($2m$)
4	Downsample($2m$)
5	Residual($4m$)
6	Downsample($4m$)
7	Residual($8m$)
8	Downsample($8m$)
9	Residual($16m$)
10	GlobalAvgPool
11	Linear($16m$)
12	LeakyRelu(0.2)
13	Linear(3)
14	Sigmoid

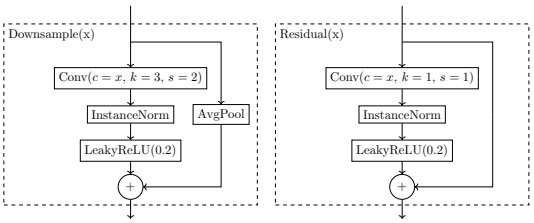


Figure 2: Architecture of the Patch Position Regression Network.

loss after 200 epochs. We manually selected a fixed patch size of $s_p = 31$ (i.e. $31 \times 31 \times 31$) voxels for all experiments. (We performed an experiment to examine the influence of the patch size on the performance (Appendix A) and found that in this setting the influence is small.) The size of the patch determines the amount of context: Thus, the amount of information the network gets, but also the sensitivity: There is a trade-off between more context, for a more accurate localization but less sensitivity to anomalies for larger patches and a greater sensitivity but inferior localization accuracy of smaller patches.

3.1. Sampling

In each iteration, we sample 256 patches from one volume. This number can be adjusted to the memory budget of the given hardware. We set this number to get a similar run-time as the baseline method (see Section 4.2). Patches that exclusively contain background (i.e. no part of the subject's anatomy) are discarded for the computation of the loss.

3.2. Dataset

For training and evaluation we used the public CQ500 dataset (Chilamkurthy et al., 2018), which contains head CT scans. For some patients, there are multiple scans present, for instance with and without contrast enhancement. Three experts determined for each volume whether there is as an intracranial haemorrhage (ICH) for each brain hemisphere and whether a cranial fracture is present. If there was a disagreement between the three raters, we used the majority vote as our ground truth. For our experiments, we used one scan from each patient without contrast enhancement, and discarded all scans that had faulty data (missing slice, wrong anatomical structure etc). The final dataset used in our experiments contains 131 images without anomaly (111 of which are used for training), and 65 images with anomalies. The details of the composition of the dataset are given in Appendix B.

3.3. Preprocessing

The volumes are resampled to voxels of size $1 \times 1 \times 1$mm with a total size of $256 \times 256 \times 256$ (that is $N = 256$) voxels. Then each volume is rigidly registered to a manually selected reference volume from the training set using AirLab (Sandkühler et al., 2018). Since CT images have a high dynamic range, we perform a histogram equalization. We segment the

skull and the two brain hemispheres in the images. This segmentation is only used for the evaluation of the method. Furthermore, we separate foreground from background to be able to filter out irrelevant patches during training.

3.4. Autoencoder Baseline

We use the basic autoencoder (AE) architecture from (Baur et al., 2021) as a baseline and adapt it to accommodate 3D volumes and to the resolution of the volumes in our dataset. The exact architecture is show in Figure 8 in Appendix C. As a post-processing step, we applied a filter (suggested in (Baur et al., 2021)) of size 5 to the reconstruction error map. We used a median filter for the fractures task, and a grayscale erosion for the ICH task. We optimized over both filter types and multiple kernel sizes to make the comparison as fair as possible.

4. Results

For comparison, we trained both our proposed method and the baseline exclusively on *normal* (healthy) data. We used the coordinate reconstruction error to detect anomalies: If the error between the actual and the predicted coordinates is high, we use that as an indication of an anomalous region. The dataset only includes labels of whether an anomaly is present in a given structure (e.g. left hemisphere). For each of these three structures we check whether the error exceeds a certain threshold, in order to predict whether an anomaly is present. Since the performance metrics of the detection, therefore, depends on this threshold, we report the *receiver operating characteristic* (ROC) and the corresponding *area under ROC* (AUROC).

4.1. Computational Resources

To evaluate the performance of our method and the baseline method with respect to the computational resources, we trained both methods with various values of the model size parameter $m = 2^0, 2^1, \ldots, 2^6$ to compare the anomaly detection performance to the size of the network. Figure 3 shows the performance as a function of the number of parameters of the networks. We can see that the performance of the networks increases up to some limit, but then decreases again. We conjecture that at a specific size, the training would benefit from more iterations or a better initialization. If we consider the best performing AE networks, we can see that the PPR network requires roughly one (fractures) or two (ICH) orders of magnitude fewer parameters to exceed the performance of the AE. For the remainder, we discuss the best performing PPR models of those that both have fewer parameters *and* used less GPU memory for the training than the best performing AE models. These are marked with an asterisk in Figure 3.

We want to point out though, that even though the memory requirements correlate with the number of parameters, they are also influenced by the actual architecture of the networks as well as the used software frameworks. Furthermore the batch size also influences the memory requirements.

In our experiments, we used batch sizes that would be used for practical purposes, that is we have $\mathrm{bs}_{exp}^{\mathrm{PPR}} = 256$ patches (sampled from one volume) for the PPR models and $\mathrm{bs}_{exp}^{\mathrm{AE}} = 4$

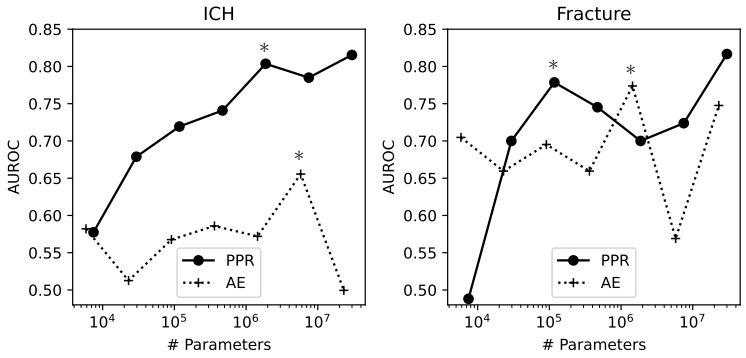


Figure 3: Performance for multiple values of the network size parameter $m = 2^0, 2^1, \ldots, 2^6$. Each of the two plots shows the performance on the test set as a function of the number of network parameters for the two types of anomalies (ICH, Fracture). The asterisk marks the best performing PPR model using fewer parameters and less GPU memory than the best performing AE model (also marked with "$*$").

Table 1: GPU Memory requirements (in MB) during training, given some batch size.

	ICH		Fracture	
Batch Size	PPR	AE	PPR	AE
bs_{exp}	4452	12548	1760	7894
1	2194	6730	1000	2742

volumes for the AE models. The GPU memory used with these settings for this is shown in the first row of Table 1, and we see that our proposed method uses about a factor of 4 less memory. But even if we only use a batch size of one for the AE models (see second row) the best performing PPR model still uses a factor of about 1.5 less memory. (The memory consumption for all our experiments is shown in Figure 10 in Appendix E.)

It should also be noted that in the ICH experiment, there were PPR models that outperformed the best performing AE model, and used even less memory during training. This illustrates, on the one hand, the general issue of the cost of handling 3D volume data and on the other hand the cost of the image reconstruction branch of the AE models that is not present in the PPR network.

4.2. Training and Testing Time requirements

We chose the batch sizes, and number of patches respectively, to result in a similar training time for all models. The time for the AE models was about 26 hours on average, the time for our PPR models is about 22 hours. While the PPR models require less memory for training, they are slower than AE models during inference, which is the price for the patch

based processing: The AE models used needed around 2 seconds to process one volume, while the PPR models needed around 1.5 minutes at the highest resolution.

4.3. Qualitative Results

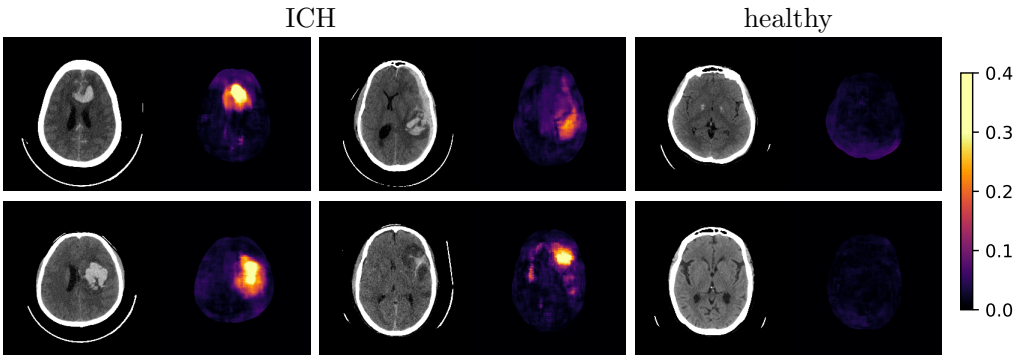


Figure 4: Slices of some selected examples that show the original CT scan with appropriate scaling of the brightness on the left, as well as the error map of our proposed PPR model (with $m = 16$). The images on the left and in the center exhibit an anomaly (ICH) while those on the right are normal (healthy).

We show some slices with examples of ICH in Figure 4 as well as surface renderings of scans of subjects with cranial fractures in Figure 5. (As a reference we also included the same examples for the AE in Appendix D in Figure 9.) It is noticeable that the reconstruction error is high where there is an anomaly. The reconstruction error generally seems to continuously depend on the amount of the patch that is anomalous, as the error maps generally seem to be rather smooth.

The performance for the models used for Figure 4 and 5 are shown in Figure 6. We observe see that the detection of fractures is the more challenging task for our method than the detection of ICH. This might be due to the smaller number of scans available to evaluate it on (see Appendix B). To put these results in context we provided a table with the inter rater agreement on these tasks in Appendix B: The performance in terms of AUROC is around $15 - 16\%$ lower than the average raters.

5. Discussion

We have shown that with our proposed approach, we can get a similar performance to AE models with a memory footprint that is up to an order of magnitude lower. This is especially interesting for processing volume data that are common in medical applications, for instance CT- or MRT scans. The lower memory requirements come at a price of a longer inference time. The time needed is still low enough for most applications in the medical field.

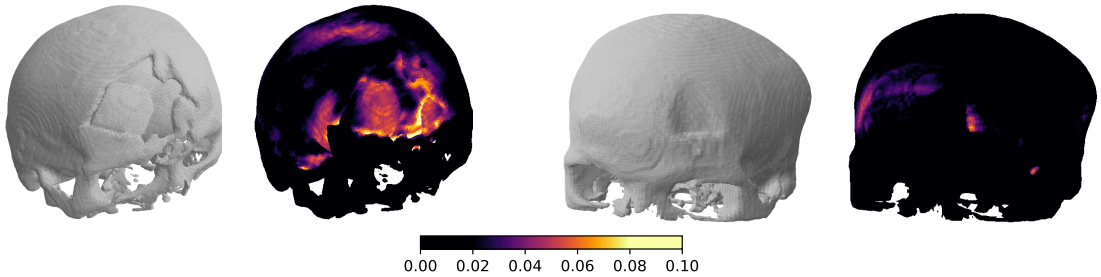


Figure 5: Surface renderings of scans of two subjects with cranial fractures. Both subjects suffer from from fractures of the frontal bone. The left side each shows the scanned part of the skull and the right part shows the same surface coloured according to the error.

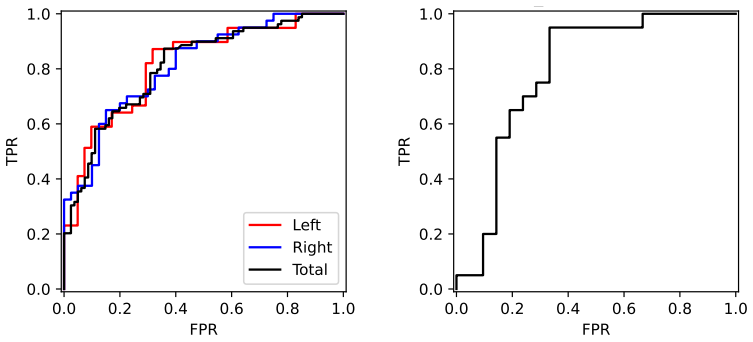


Figure 6: Left: ROC curve for the detection of ICH (AUROC = 0.81), Right: ROC curve for cranial fractures (AUROC = 0.79).

We see the proposed network as a proof-of-concept that would be interesting for further investigation. One of the drawbacks of this method is the limited spatial resolution. This could be addressed with a multi stage coarse-to-fine approach similar to what was proposed in (Lei et al., 2021).

Furthermore, it would be interesting to investigate the influence of the patch size in relation to the spatial extent of the anomalies, and to see if it would be possible to combine it with a tissue classification task. In addition to the patch size it would also be to vary the shape of the patch that is passed into the network

Acknowledgments

We are grateful for the support of the Novartis FreeNovation initiative and the Uniscientia Foundation (project #147-2018). We would also like to thank the NVIDIA Corporation for the donation of a GPU that was used for our experiments.

References

Muhammad S. Battikh and Artem Lenskiy. Latent-insensitive autoencoders for anomaly detection and class-incremental learning. *ArXiv*, abs/2110.13101, 2021.

Christoph Baur, Stefan Denner, Benedikt Wiestler, Nassir Navab, and Shadi Albarqouni. Autoencoders for unsupervised anomaly segmentation in brain MR images: A comparative study. *Medical Image Analysis*, 69:101952, 2021. ISSN 1361-8415.

Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain MRI using constrained adversarial auto-encoders. In *Medical Imaging with Deep Learning (MIDL) 2018*, 2018.

Sasank Chilamkurthy, Rohit Ghosh, Swetha Tanamala, Mustafa Biviji, Norbert G Campeau, Vasantha Kumar Venugopal, Vidur Mahajan, Pooja Rao, and Prashant Warier. Deep learning algorithms for detection of critical findings in head CT scans: a retrospective study. *The Lancet*, 392(10162):2388–2396, 2018.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Wenhui Lei, Wei Xu, Ran Gu, Hao Fu, Shaoting Zhang, Shichuan Zhang, and Guotai Wang. Contrastive learning of relative position regression for one-shot object localization in 3D medical images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 155–165. Springer, 2021.

Daisuke Sato M.D., Shouhei Hanaoka M.D., Yukihiro Nomura, Tomomi Takenaga, Soichiro Miki M.D., Takeharu Yoshikawa M.D., Naoto Hayashi M.D., and Osamu Abe M.D. A primitive study on unsupervised anomaly detection with an autoencoder in emergency head ct volumes. In Nicholas Petrick and Kensaku Mori, editors, *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 388 – 393. International Society for Optics and Photonics, SPIE, 2018. doi: 10.1117/12.2292276. URL https://doi.org/10.1117/12.2292276.

Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks. *arXiv preprint arXiv:1801.07372*, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

Robin Sandkühler, Christoph Jud, Simon Andermatt, and Philippe C Cattin. AirLab: autograd image registration laboratory. *arXiv preprint arXiv:1806.09907*, 2018.

Alexander Tong, Guy Wolf, and Smita Krishnaswamy. Fixing bias in reconstruction-based anomaly detection with Lipschitz discriminators. *Journal of Signal Processing Systems*, pages 1–15, 2021.

Julia Wolleb, Robin Sandkühler, and Philippe C Cattin. DeScarGAN: Disease-specific anomaly detection with weak supervision. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 14–24. Springer, 2020.

Appendix A. Patch Size

To examine the influence of the patch size (see Section 3) we evaluated the ICH task with identical settings but different patch sizes. For each of the patch sizes we trained a model from scratch with the same configuration as in the other experiments. In Figure 7 we show the AUROC score as a function of the patch size for the detection of the haemorrhages in each brain hemisphere (red and blue) as well as the total score for both hemispheres combined. Across all six patch sizes the performance slightly changes by about ± 0.06. Comparing that value with the variation of the individual brain hemispheres we conclude that the influence of the patch size in these experiments in negligible. For other anomalies or modalities this might be different though.

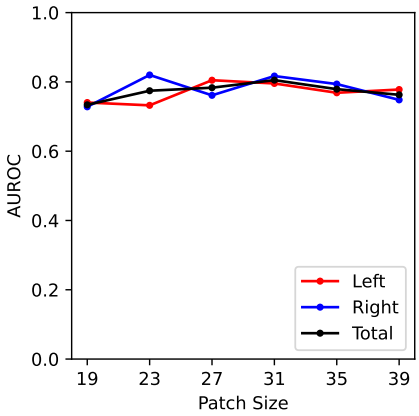


Figure 7: Performance of the PPR network in terms of AUROC as a function of the patch size $s_p = 19, 23, 27, 31, 35, 39$ for the ICH task.

Appendix B. CQ500 Dataset

The scans with bleeding or a fracture were considered *anomalous*, while the scans without any of these findings were considered *healthy*. The test set contains 86 volumes in total, including 21 from the healthy set (i.e. without anomalies), 20 with a fracture, 39 with

bleeding in the left hemisphere and 47 with bleeding in the right hemisphere. Note that these sets are not disjoint, Table 2 shows the actual distribution. Our test set consists of the anomalous scans as well as 21 randomly chosen scans from the healthy set. The remaining scans of the healthy set were used for training.

Table 2: Distribution of the anomalous data in the test set. Overlined column/row names indicate the absence of the given feature. Each entry shows the number of volumes with that combination of the presence/absence of these three features.

	Bleeding Left		$\overline{\text{Bleeding Left}}$	
	Bleeding Right	$\overline{\text{Bleeding Right}}$	Bleeding Right	$\overline{\text{Bleeding Right}}$
Fracture	4	3	7	6
$\overline{\text{Fracture}}$	16	16	13	21

To characterize the variability of the ground truth within the three raters, we compute the Fleiss-Kappa as well as the pairwise AUROC in Table 3.

Table 3: Inter rater agreement expressed using the Fleiss-Kappa as well as AUROC for each rater compared to the majority vote for each feature.

	κ	AUROC			
		R1	R2	R2	AVG
Bleeding Left	0.745	0.877	0.982	0.985	0.948
Bleeding Right	0.705	0.893	0.945	0.964	0.934
Fracture	0.632	0.901	0.955	0.921	0.926

Appendix C. Autoencoder Baseline

We trained the AE network for 2000 epochs with the Adam optimizer with $\text{lr} = 0.001$. The architecture of the AE is shown in Figure 8: The architecture is parametrized by m to be able to consider networks of different sizes. We define two blocks, "Downsample" and "Upsample", that are used throughout the network. "[Transp]Conv(c, k, s)" stands for a 3D [transposed] convolution with c output channels, a kernel size of k and a stride of s. The AE models were trained using a pixel-wise L_1-loss.

#	**Block**
1	Downsample(m)
2	Downsample($2m$)
3	Downsample($4m$)
4	Downsample($8m$)
5	Downsample($16m$)
6	Upsample($8m$)
7	Upsample($4m$)
8	Upsample($2m$)
9	TranspConv($c = 1, k = 4, s = 2$)
10	Sigmoid

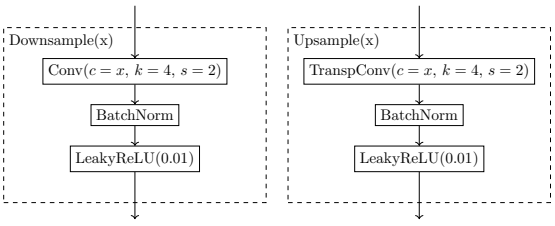


Figure 8: Architecture of the AE network.

Appendix D. Qualitative Results AE

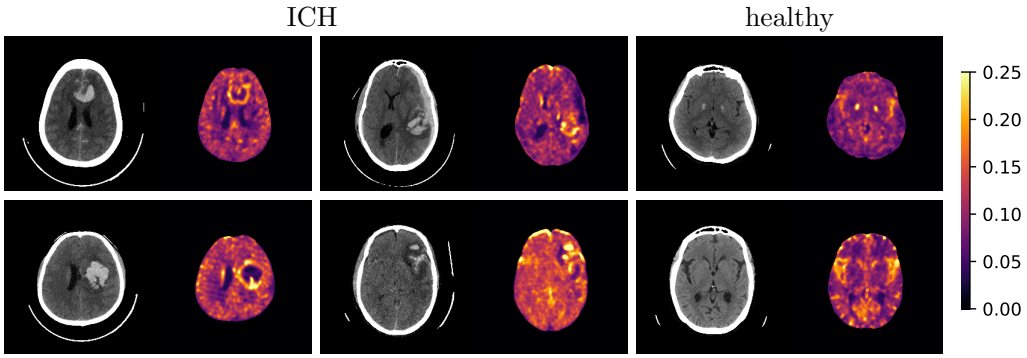


Figure 9: Slices of some selected examples that show the original CT scan with appropriate scaling of the brightness on the left, as well as the error map the baseline AE method. The images on the left and in the center exhibit an anomaly (ICH) while those on the right are normal (healthy).

Appendix E. Memory Consumption

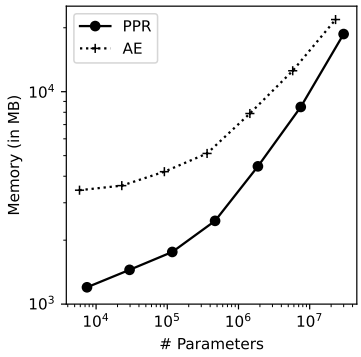


Figure 10: GPU Memory consumption as a function of the number of parameters for the AE and PPR with network size parameter $m = 2^0, 2^1, \ldots, 2^6$ and batch sizes bs_{exp} as used in the experiments.

Chapter 7

Memory-Efficient 3D Denoising Diffusion Models for Medical Image Processing

In this paper, we try to enable the use of denoising diffusion models for 3D data. Specifically, we build on a segmentation method[140] that was previously proposed for 2D data. We propose techniques to make the underlying model more memory efficient, and enable it to be applied to 3D data, in this case MR scans of the BraTS challenge[25]. With these techniques, we enable a significant reduction in resource consumption and therefore the processing of high resolution 3D volumes using denoising diffusion models.

Publication. The following paper was presented at the *Medical Imaging with deep Learning* (MIDL) Conference, July 2023, Nashville, TN, USA. It was published as part of the conference proceedings [92].

Proceedings of Machine Learning Research 227:552–567, 2023 MIDL 2023 – Full paper track

Memory-Efficient 3D Denoising Diffusion Models for Medical Image Processing

Florentin Bieder[1] FLORENTIN.BIEDER@UNIBAS.CH
Julia Wolleb[1] JULIA.WOLLEB@UNIBAS.CH
Alicia Durrer[1] ALICIA.DURRER@UNIBAS.CH
Robin Sandkühler[1] ROBIN.SANDKUEHLER@UNIBAS.CH
Philippe C. Cattin[1] PHILIPPE.CATTIN@UNIBAS.CH
[1] *Department of Biomedical Engineering, University of Basel*

Abstract

Denoising diffusion models have recently achieved state-of-the-art performance in many image-generation tasks. They do, however, require a large amount of computational resources. This limits their application to medical tasks, where we often deal with large 3D volumes, like high-resolution three-dimensional data. In this work, we present a number of different ways to reduce the resource consumption for 3D diffusion models and apply them to a dataset of 3D images. The main contribution of this paper is the memory-efficient patch-based diffusion model *PatchDDM*, which can be applied to the total volume during inference while the training is performed only on patches. While the proposed diffusion model can be applied to any image generation task, we evaluate the method on the tumor segmentation task of the BraTS2020 dataset and demonstrate that we can generate meaningful three-dimensional segmentations.

Keywords: diffusion models, three-dimensional, supervised segmentation

1. Introduction

Denoising diffusion models (Ho et al., 2020; Nichol and Dhariwal, 2021) have lately shown an impressive performance in image generation and experienced increasing popularity in medical image analysis (Kazerouni et al., 2022). However, the processing of large three-dimensional (3D) volumes, which often is required in medical applications, is still a challenge. Limitations related to the computational resources only allow the processing of small 3D volumes, which impedes the processing of high-resolution magnetic resonance (MR) or computer tomography (CT) scans.

Contribution In this work, we introduce architectural changes to the state-of-the-art diffusion model implementation (Nichol and Dhariwal, 2021), enabling to train on large 3D volumes with commonly available GPUs. We adapt the U-Net-like architecture to improve the speed and memory efficiency. Furthermore, we propose a novel method illustrated in Figure 1. With this method, the diffusion model is trained only on coordinate-encoded patches of the input volume, which reduces the memory consumption and speeds up the training process. During sampling, the proposed method allows processing large volumes in their full resolution without needing to split them up into patches. To evaluate our method, we perform diffusion model based image segmentation (Wolleb et al., 2022b) that has previously been proposed for 2D segmentation on the BraTS2020 dataset (Menze et al., 2014; Bakas et al., 2017, 2018). The code is available at github.com/florentinbieder/PatchDDM-3D.

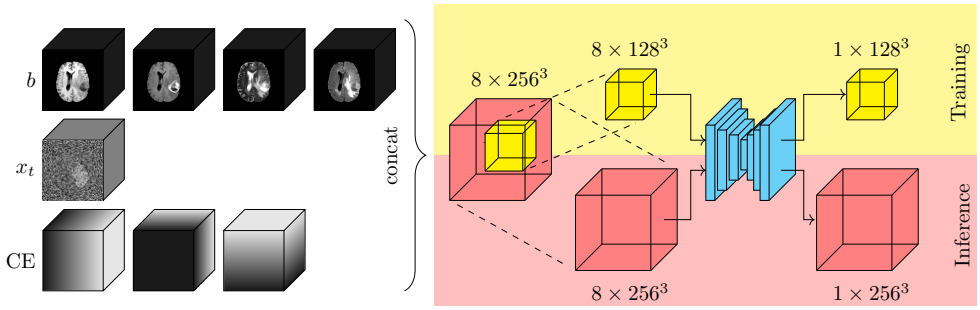


Figure 1: Overview of our proposed method *PatchDDM*. The diffusion model is optimized in memory efficiency and speed by training only on coordinate-encoded patches. The input consists of noised x_t, the volumes b that are to be segmented and which are provided as a condition for the segmentation, as well as a coordinate encoding CE for the patches. During sampling, the whole 3D volume can be processed at once.

Related Work Denoising diffusion models have seen a quick adoption in research, replacing the more traditional generative models in many tasks such as such as unconditional and conditional image generation (Ho et al., 2020; Song et al., 2021; Nichol and Dhariwal, 2021), text-to-image translation (Nichol et al., 2021; Saharia et al., 2022b; Ramesh et al., 2021; Kim et al., 2022) and inpainting (Ramesh et al., 2021; Nichol et al., 2021). Diffusion models have also been used for various applications in the medical field, for instance, for anomaly detection (Wolleb et al., 2022a), synthetic image generation (Dorjsembe et al., 2022; Peng et al., 2022) and segmentation (Guo et al., 2022; Wu et al., 2022; Wolleb et al., 2022b). Medical images, however, often are 3D volumes, such as MR- or CT-scans. These volumes create challenges regarding the memory consumption of processing methods. Consequently, many of the current methods are limited to two-dimensional (2D) slices only (Wu et al., 2022; Wolleb et al., 2022b; Guo et al., 2022) or to 3D volumes restricted to a limited resolution of at most $128 \times 128 \times 128$ (Khader et al., 2022; Peng et al., 2022; Dorjsembe et al., 2022). To the best of our knowledge, we are the first to tackle the challenge of applying denoising diffusion models to large 3D volumes.

2. Method

We explore how denoising diffusion implicit models (DDIMs) presented in Section 2.1 can be improved regarding memory efficiency and time consumption. The required architectural changes are presented in Section 2.2. We present the training and sampling scheme of our method *PatchDDM* in Section 2.3. For evaluation, we use the segmentation approach using denoising diffusion models presented in Section 2.4.

2.1. Denoising Diffusion Models

In the following, we will use the notation introduced by (Ho et al., 2020). Denoising diffusion models rely on an iterative noising and denoising process. The forward noising process q is given by

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$

where β_t is a predefined sequence of variances. We can directly compute x_t from a given x_0 with

$$q(x_t \mid x_0) := \mathcal{N}(x_t; \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})I) \tag{2}$$

with $\alpha_t := 1 - \beta_t$ and $\overline{\alpha_t} := \prod_{s=1}^{t} \alpha_s$. This corresponds to degrading the input image by adding Gaussian noise. For image generation tasks we are interested in the reverse process p_ϑ.

$$p_{\vartheta,t}(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_{\vartheta,t}(x_t), \Sigma_{\vartheta,t}(x_t)) \tag{3}$$

Both $\mu_{\vartheta,t}$ and $\Sigma_{\vartheta,t}$ can be estimated by a U-Net-based network $\varepsilon_{\vartheta,t}$ with parameters ϑ. The loss used to train the network $\varepsilon_{\vartheta,t}$ can be written as

$$\|\varepsilon - \varepsilon_{\vartheta,t}(x_t, t)\|^2 = \|\varepsilon - \varepsilon_{\vartheta,t}(\sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\varepsilon, t)\|^2, \quad \text{with } \varepsilon \sim \mathcal{N}(0, I). \tag{4}$$

Using the DDIM (Song et al., 2021) sampling scheme, we can define

$$x_{t-1} = \sqrt{\overline{\alpha_{t-1}}}\left(\frac{x_t - \sqrt{1-\overline{\alpha_t}}\varepsilon_{\vartheta,t}(x_t)}{\sqrt{\overline{\alpha_t}}}\right) + \sqrt{1-\overline{\alpha_{t-1}}}\varepsilon_{\vartheta,t}(x_t), \tag{5}$$

where $\varepsilon_\vartheta(x_t)$ is the output of the network. This sampling scheme has the advantage that the denoising process is deterministic and we do not need to sample random vectors in every step. Thus, the only source of stochasticity during inference originates from the random initial sample x_T which is sampled from $\mathcal{N}(0, I)$. During inference, a sequence of images x_i for $i = T, T-1, \ldots, 0$ of decreasing noise level is being generated, the initial x_T is sampled from a standard normal distribution $\mathcal{N}(0, I)$.

2.2. Architecture

We adapt the 2D-U-Net-based network architecture proposed by (Ho et al., 2020; Nichol and Dhariwal, 2021) and used by (Nichol et al., 2021; Bansal et al., 2022; Wu et al., 2022; Song et al., 2021) for the application on 3D data. The previously proposed architecture features two or three residual convolutional blocks at each down- and upsampling step. Furthermore, it uses attention blocks at multiple resolutions as well as in the bottleneck. (Saharia et al., 2022a) determined that adding global self attention can slightly improve the quality of the generated images as compared to an increase in convolutional blocks. For 3D data, the attention blocks use disproportionally more memory, which made it infeasible to use them on current hardware, which is why we removed them completely. The second fundamental change we implemented was the use of additive skip connections, as shown in Figure 2. In the previous architecture as well as in the original U-Net implementation (Ronneberger et al., 2015), the skip connection uses concatenation to combine x_s from the encoder with the upsampled tensors x_u from the lower resolution path of the decoder. This

PatchDDM

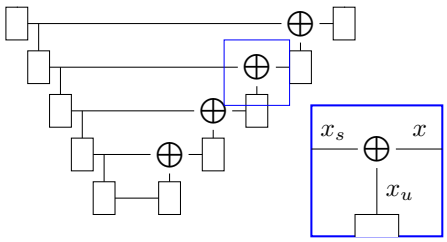


Figure 2: The architecture of the U-Net-like network with averaging skip connections. In the original network as well as in the U-Net the $\bigoplus$ operator is a concatenation $x = (x_s, x_u)$, in our case it is an averaging operator $x = (x_s + x_u)/2$.

implies that the decoder requires significantly more resources than the encoder, especially at the highest resolution levels.

To alleviate this issue, we propose to average them as $x = \frac{1}{f}(x_s + x_u)$ with $f = 2$. Unlike in ResNet (He et al., 2016), where the skip connections are added ($f = 1$), we found the averaging to be crucial for avoiding numerical issues like exploding gradients. Intuitively this can be justified by considering x_s and x_u as random variables with x_u, x_s iid. $\mathcal{N}(0, \sigma^2)$. Therefore, $\frac{1}{f}(x_u + x_s) \sim \mathcal{N}(0, \frac{2}{f^2}\sigma^2)$. This means that for $f = 1$ (the summation as in ResNet), each concatenation the variance doubles. To prevent the variance from increasing should chose $f \geqslant \sqrt{2}$. We chose heuristically $f = 2$, i.e. averaging.

The savings in memory from replacing the concatenation with the averaging allow us to increase the network width, i.e. the number of channels within the whole network, by a factor of 1.61, while preserving the total memory usage.

Furthermore, the resulting network architecture allows for training on varying input sizes. This property is crucial for our proposed patch-based method. For all of our experiments, we use the same network configuration.

2.3. Patch-based Approach with Coordinate-encoding

To benefit from the lower requirements of computational resources but still to operate on the original resolution, we propose a novel patch-based training method named *PatchDDM* that trains on randomly sampled patches of the input but can afterwards be applied to the full resolution volume during inference. This means for the training we can benefit from using smaller inputs, which means we need fewer computations per iteration as well as less memory. For the inference, however, we can pass the entire input volume at once *without* having to sample patches and reassemble them. This means no boundary artifacts due to the separate padding of the patches within the CNN are introduced, and also the stitching artifacts that that can appear in traditional patch-based approaches are eliminated.

To add information about the position of the patch, we condition the network on the position of the sampled patch. We implemented this by concatenating a grid of Cartesian coordinates to the input. Each coordinate is represented by one channel as a linear gradient

ranging from -1 to 1. This is similar to the method proposed in (Liu et al., 2018). They propose to add the coordinates as additional channels before all convolutions.

In our case, we append the coordinates to the whole input just like in (Liu et al., 2018), but then sample a patch, where the coordinates serve as a position encoding for the sampled patch. An overview of this coordinate encoding is given in Figure 1. For the BraTS2020 data, the subject is centered within the volume. We use a patch sampling strategy, assigning a higher probability to the center of the volume, as shown in Figure 6 in the Appendix A.

Baseline methods For our ablation study, we use two baselines with the same network as our proposed approach, but without patch-based training. Furthermore, we also performed an experiment with our patch based approach but without the proposed coordinate encoding. The training did not converge and did not produce any usable results. Therefore, we will not report any metrics from this experiment. The two baseline methods are the following:

- Training on full resolution (*FullRes*): We implemented a distributed version of the proposed architecture that splits the task to two GPUs if necessary. This allows for training directly on full resolution (256^3) data, given that the expensive specialized GPU hardware is available.

- Training on half resolution (*HalfRes*): A straightforward way to reduce the requirements in terms of computational resources is training the model on downsampled data. In our experiments, we downsampled the input image before passing it to the network, but then upsampled the output of the network again to evaluate the performance on the full size. For three spatial dimensions (i.e. 3D) this means that reducing the input size from 256^3 to 128^3 results in a reduction of a factor of 8 in terms of memory and computation time, allowing this model to be run on widely available GPUs.

2.4. Denoising Diffusion Models with Ensembling for Segmentation

In order to generate the segmentation of an input image b, we need to condition the generation of the segmentation mask x_0 on that given image b. We will follow the method proposed by (Wolleb et al., 2022b), where the input images b are being concatenated to every x_t as a condition. It was shown that ensembling several predicted segmentation masks per input image increases the segmentation performance (Amit et al., 2021; Wolleb et al., 2022b). An overview of this segmentation approach is given in Figure 3. An advantage of the denoising diffusion based segmentation approach is the implicit ensembling we get when using different samples x_T from the noise distribution $\mathcal{N}(0, I)$, which can be used to increase the performance and estimate the uncertainty. To evaluate the performance of our proposed method *PatchDDM* described in Section 2.3 and the two baseline methods *FullRes* and *HalfRes*, we apply our method to a segmentation task as proposed in (Wolleb et al., 2022b). Therefore, we train our diffusion model to generate semantic segmentation masks.

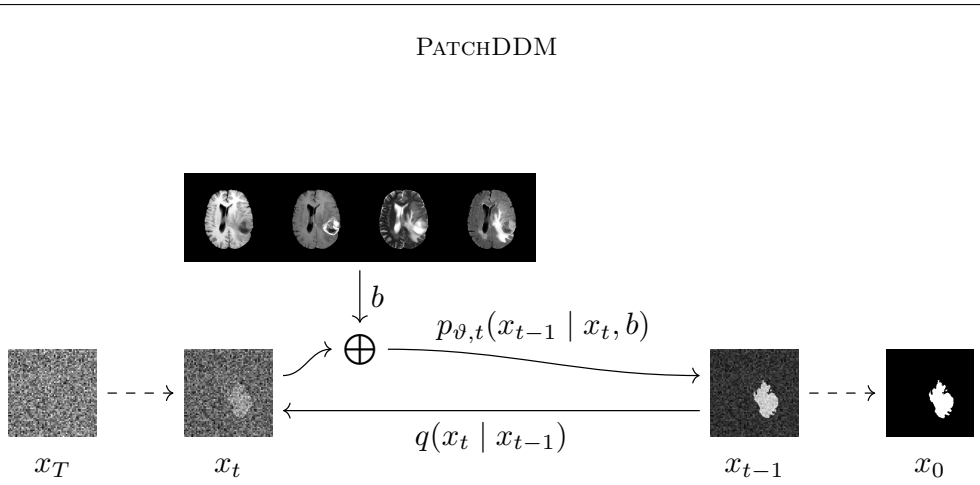


Figure 3: The ground truth segmentation x_0 is degraded by the noising process q. We train a network to perform the denoising process p_ϑ, that is, given some noised image x_t, we train it to denoise it with the MR-sequences b as a condition.

3. Experiments

Dataset For our experiments, we used the BraTS2020 dataset (Menze et al., 2014; Bakas et al., 2017, 2018). It contains 369 head MR-scans, each including four sequences (T1, T1ce, T2, FLAIR) with a resolution of $1 \times 1 \times 1$ mm^3, resulting in a total scan size of $240 \times 240 \times 155$, which we padded to a size of $256 \times 256 \times 256$. The background voxels were set to zero and the range between the first and 99th percentile was normalized to $[0, 1]$. We used an 80%/10%/10% split for training, validation and testing. The label masks consist of three classes, namely the Gadolinium-enhancing tumor, the peritumoral edema, and the necrotic and non-enhancing tumor core. For the binary segmentation experiments, all three classes were merged into one.

Training Details We performed our experiments on NVIDIA A100 GPUs with $40GB$ of memory each. To directly train on the full resolution 256^3 images, we distribute the model over 2 GPUs. The methods *HalfRes* and *PatchDDM* were trained on one GPU only. The optimizer we used was AdamW (Loshchilov and Hutter, 2017) with the default parameters. We chose the learning rate $\mathrm{lr} = 10^{-5}$ by optimizing the average Dice coefficient on the validation set after 150k optimization steps over a range of values between 10^{-6} and 10^{-3}. We trained the models for the same amount of time for all experiments (420h). For the evaluation, we selected the best-performing models based on the average Dice score on the validation set based on a single evaluation, i.e., without ensembling. For the denoising process, we set the number of steps to $T = 1000$ and use the affine variance schedule proposed in (Ho et al., 2020) with $\beta_1 = 0.02, \beta_T = 10^{-4}$.

Accelerated Sampling By default, we need $T = 1000$ denoising steps for the inference. As shown in (Song et al., 2021), we can interpret the the DDIM denoising step (5) as the Euler discretization of an ordinary differential equation (ODE).

This insight motivates the use of larger step sizes with respect to t during inference, which allows for accelerated sampling. The drawback is that the output quality deteriorates

with fewer samples. We investigate how we can trade off fewer sampling steps (larger step sizes) and ensembling (more samples).

4. Results

In the following, we will assess the performance of our proposed model and compare it to the two baseline approaches. For each model, we computed the average Dice score on the validation set and used this to choose the best-performing checkpoint. We provide some qualitative outputs in Figure 7 in the Appendix B.

To assess the training progress, we display the Dice score as well as the HD95 (Hausdorff distance, 95th percentile) of *PatchDDM* over the course of the training in Figure 8 in the Appendix C. The metrics of the best-performing checkpoint with respect to the Dice score when using a single evaluation (no ensembling) is reported in Table 2 in Appendix D along with the score of the state of the art nnU-Net (Isensee et al., 2021).

4.1. Segmentation Ensembling

To evaluate the impact of ensembling, we compute the Dice- and HD95-score of the three methods (*PatchDDM*, *FullRes*, *HalfRes*) with respect to ensemble size, see Figure 4. Both scores significantly improve using ensembles for our proposed *PatchDDM* and the *FullRes* method. In Table 3 in Appendix E the metrics for different ensemble sizes are provided. The curves show that ensembling can further improve the performance and get very close to the best performing ensembles with an ensemble size of as small as five to nine.

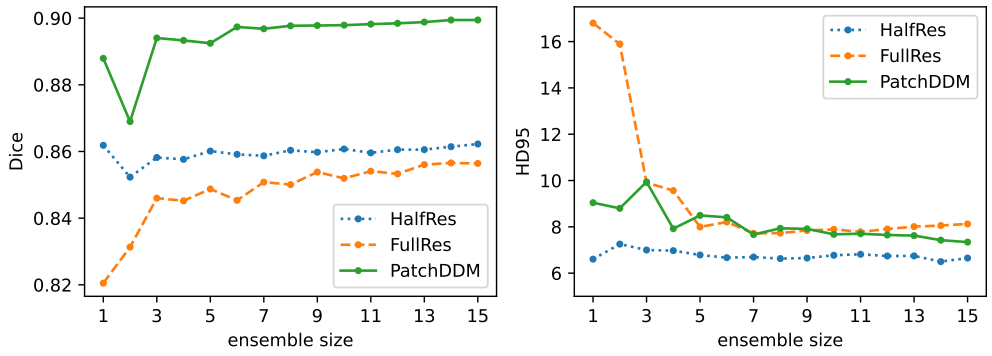


Figure 4: The evaluation metrics on the test set as a function of the ensemble size.

4.2. Computational Resources & Time Requirements

We report the memory consumption and the time required for one model evaluation for all comparing methods. As displayed in Table 1, the training of *FullRes* needs close to 80GB of memory. This requires at the time of writing still highly expensive hardware. The other baseline *HalfRes* as well as our proposed *PatchDDM* method both need less than 12GB for training and can therefore be trained on much cheaper and widely available hardware. The

reduced resolution also results in a reduction in the number of computations, and therefore a larger number of optimization steps that can be performed in a given time interval. A drawback of our proposed method is the increased memory consumption and reduced speed during inference, both of which are comparable to the *FullRes* model.

Table 1: Memory consumption in GB and time in seconds for one network evaluation. The memory requirements for the distributed run also include a small amount of overhead, as some arrays are duplicated on both GPUs.

	Memory		**Time**	
Method	**Training**	**Inference**	**Training**	**Inference**
FullRes	78.5	25.7	2.12	1.01
HalfRes	10.5	4.90	0.351	0.124
PatchDDM	10.6	24.0	0.340	1.02

4.3. Ensembling and Accelerated Sampling

Figure 5 shows the trade-off between the ensemble size and the number of sampling steps. With as little as 20 sampling steps (i.e. a step size of 50), the performance is already close to the results obtained with $T = 1000$ steps, implying a speedup of a factor of 50. But even with fewer step sizes, we can trade the number of steps for a greater ensemble size to achieve a similar performance. Consequently, for a fixed budget of network evaluations (i.e. steps), we can profit from using ensembling with accelerated sampling.

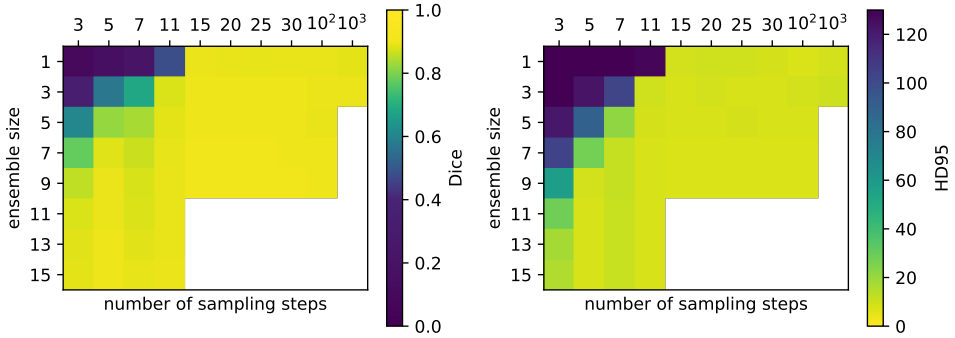


Figure 5: The average Dice score and HD95 metric on the test set as a function of the number of sampling steps and the ensemble size. The white sections indicate that we did not evaluate that combination.

5. Discussion

We propose *PatchDDM*, a novel patch-based diffusion model architecture that allows the training of diffusion models on high-resolution 3D datasets. This enables denoising diffusion models to be used for image analysis and -processing tasks in medicine on commonly available hardware. We could demonstrate the effectiveness by applying it to a recently developed segmentation framework for medical images. In the future, we would like to investigate the performance of our proposed approach for tasks involving image generation. Furthermore, we will investigate the role of the patch size used and whether it can be made smaller for processing even higher resolution volumes. In order to preserve high quality, (Karras et al., 2022) proposed using higher-order ODE solvers, like the Heun method, when choosing larger step sizes. This might further reduce the number of iterations needed. Finally, it would be interesting to investigate an extension of this segmentation framework that includes multiple classes.

Acknowledgments

We are grateful for the support of the Novartis FreeNovation initiative and the Uniscientia Foundation (project #147-2018). We would also like to thank the NVIDIA Corporation for donating a GPU that was used for our experiments.

References

Tomer Amit, Eliya Nachmani, Tal Shaharbany, and Lior Wolf. Segdiff: Image segmentation with diffusion probabilistic models. *arXiv preprint arXiv:2112.00390*, 2021.

Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, Justin S Kirby, John B Freymann, Keyvan Farahani, and Christos Davatzikos. Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific data*, 4(1):1–13, 2017.

Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Takeshi Shinohara, Christoph Berger, Sung Min Ha, Martin Rozycki, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge. *arXiv preprint arXiv:1811.02629*, 2018.

Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise. arXiv, 2022. doi: 10.48550/ARXIV.2208.09392.

Zolnamar Dorjsembe, Sodtavilan Odonchimed, and Furen Xiao. Three-Dimensional Medical Image Synthesis with Denoising Diffusion Probabilistic Models. In *Medical Imaging with Deep Learning*, 2022.

Xutao Guo, Yanwu Yang, Chenfei Ye, Shang Lu, Yang Xiang, and Ting Ma. Accelerating Diffusion Models via Pre-segmentation Diffusion Sampling for Medical Image Segmentation. *arXiv preprint arXiv:2210.17408*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. *arXiv preprint arXiv:2206.00364*, 2022.

Amirhossein Kazerouni, Ehsan Khodapanah Aghdam, Moein Heidari, Reza Azad, Mohsen Fayyaz, Ilker Hacihaliloglu, and Dorit Merhof. Diffusion models for medical image analysis: A comprehensive survey. *arXiv preprint arXiv:2211.07804*, 2022.

Firas Khader, Gustav Mueller-Franzes, Soroosh Tayebi Arasteh, Tianyu Han, Christoph Haarburger, Maximilian Schulze-Hagen, Philipp Schad, Sandy Engelhardt, Bettina Baessler, Sebastian Foersch, et al. Medical Diffusion–Denoising Diffusion Probabilistic Models for 3D Medical Image Generation. *arXiv preprint arXiv:2211.03364*, 2022.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2426–2435, June 2022.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021.

Wei Peng, Ehsan Adeli, Qingyu Zhao, and Kilian M Pohl. Generating Realistic 3D Brain MRIs Using a Conditional Diffusion Probabilistic Model. *arXiv preprint arXiv:2212.08034*, 2022.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

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

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-Image Diffusion Models. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450393379. doi: 10.1145/3528233.3530757.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *arXiv preprint arXiv:2205.11487*, 2022b.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021.

Julia Wolleb, Florentin Bieder, Robin Sandkühler, and Philippe C Cattin. Diffusion Models for Medical Anomaly Detection. *arXiv preprint arXiv:2203.04306*, 2022a.

Julia Wolleb, Robin Sandkuehler, Florentin Bieder, Philippe Valmaggia, and Philippe C. Cattin. Diffusion Models for Implicit Image Segmentation Ensembles. In *Medical Imaging with Deep Learning*. Proceedings of Machine Learning Research, 2022b.

Junde Wu, Huihui Fang, Yu Zhang, Yehui Yang, and Yanwu Xu. MedSegDiff: Medical Image Segmentation with Diffusion Probabilistic Model. *arXiv preprint arXiv:2211.00611*, 2022.

Appendix A. Sampling Distribution

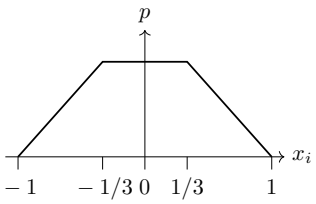


Figure 6: The sampling distribution was chosen empirically to favor the central patches. The distribution p is defined over the normalized coordinates of admissible patches (normalized to $[-1, 1]$) and can be interpreted as the probability density function the sum $X + Y$ of two random variables $X \sim U[-1/3, 1/3], Y \sim U[-2/3, 2/3]$.

Appendix B. Qualitative Results

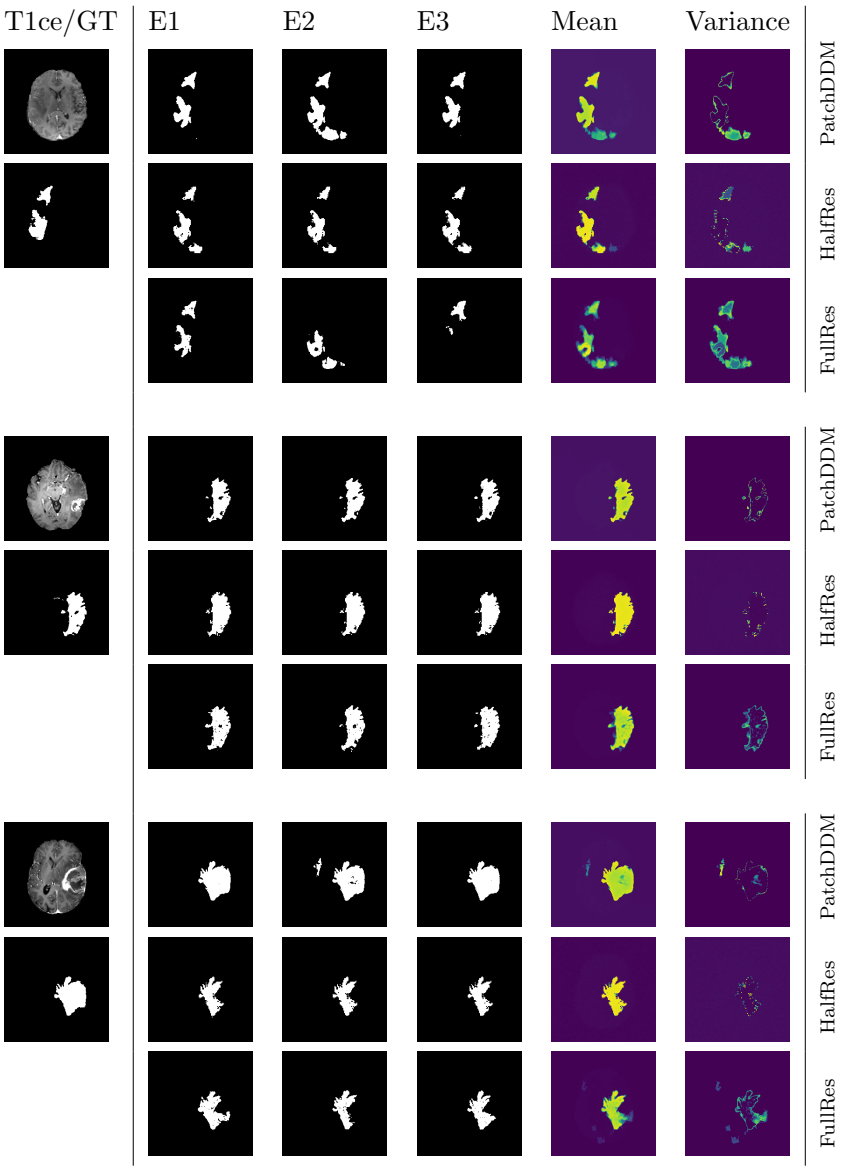


Figure 7: We display an axial slice of three volumes. The first column shows T1ce-sequence and the ground truth segmentation. Then we display three outputs E1-E3 of the ensemble for each of the models and finally the mean- and (normalized) variance map across the ensemble of size 15.

Appendix C. Training Progress

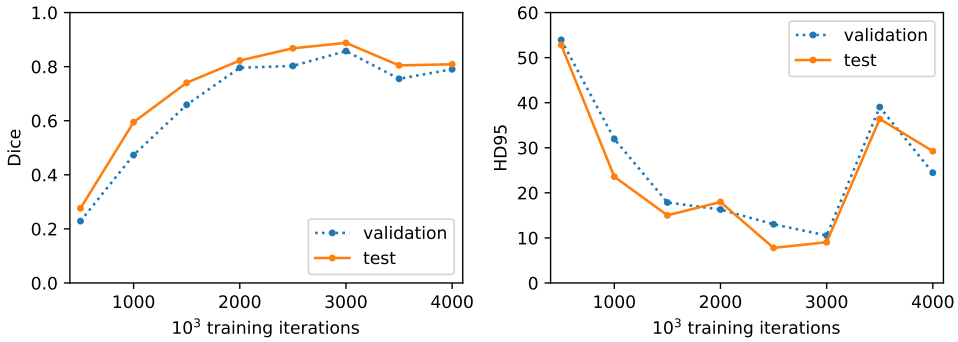


Figure 8: Performance of our method *PatchDDM* on the validation- and test set over the course of the training. The x-axis indicates the number of training iterations as a multiple of 1000.

Appendix D. Single Evaluation Scores

Table 2: Segmentation scores of our methods and nnU-Net on different metrics on our test set based on a single evaluation.

Method	**Dice**	**HD95**
FullRes	0.82 ± 0.12	16.80 ± 18.96
HalfRes	0.86 ± 0.09	6.61 ± 9.37
PatchDDM	0.88 ± 0.07	9.04 ± 8.75
nnU-Net	0.96 ± 0.02	1.24 ± 0.48

Appendix E. Ensembling Scores

Table 3: Segmentation scores of the three methods with various ensemble sizes.

Method	Ensemble size	1	3	5	7	15
FullRes	Dice	0.821	0.846	0.849	0.851	0.856
	HD95	16.80	9.91	8.00	7.72	8.13
HalfRes	Dice	0.862	0.858	0.860	0.859	0.862
	HD95	6.61	7.00	6.79	6.70	6.65
PatchDDM	Dice	0.888	0.894	0.892	0.897	0.899
	HD95	9.04	9.94	8.49	7.67	7.34

Chapter 8

Modeling the Neonatal Brain Development using Implicit Neural Representations

This work explores the use of implicit neural representations for modelling the brain development process of neonates. Using data of the developing Human Connectome Project (dHCP) [24, 23], which includes head MR scans of neonates along with their age, we try to model the development of the brains between 26 and 45 weeks of postmenstrual age.

Publication. The following paper will be presented at the *Predictive Intelligence in Medicine* Workshop (PRIME) at the *Medical Image Computing and Computer Assisted Intervention* (MICCAI) Conference, October 2024, Marrakesh. It will be published as part of the conference proceedings.

Modeling the Neonatal Brain Development Using Implicit Neural Representations

Florentin Bieder[1] [0000−0001−9558−0623] florentin.bieder@unibas.ch,
Paul Friedrich[1] [0000−0003−3653−5624] paul.friedrich@unibas.ch,
Hélène Corbaz[1] [0009−0006−9716−5136] helene.corbaz@unibas.ch,
Alicia Durrer[1] [0009−0007−8970−909X] alicia.durrer@unibas.ch,
Julia Wolleb[1] [0000−0003−4087−5920] julia.wolleb@unibas.ch,
and Philippe C. Cattin[1] [0000−0001−8785−2713] philippe.cattin@unibas.ch

Department of Biomedical Engineering, University of Basel

Abstract. The human brain undergoes rapid development during the third trimester of pregnancy. In this work, we model the neonatal development of the infant brain in this age range. As a basis, we use MR images of preterm- and term-birth neonates from the developing human connectome project (dHCP). We propose a neural network, specifically an implicit neural representation (INR), to predict 2D- and 3D images of varying time points. In order to model a subject-specific development process, it is necessary to disentangle the age from the subjects' identity in the latent space of the INR. We propose two methods, Subject Specific Latent Vectors (SSL) and Stochastic Global Latent Augmentation (SGLA), enabling this disentanglement. We perform an analysis of the results and compare our proposed model to an age-conditioned denoising diffusion model as a baseline. We also show that our method can be applied in a memory-efficient way, which is especially important for 3D data.

Keywords: implicit neural representations, neonatal development, MRI

1 Introduction

The development of the central nervous system begins early in pregnancy and can be detected using ultrasound as early as eight weeks of gestation [14]. At around 20 weeks, the corpus callosum is fully developed, and the first structures of the cortical surface begin to form [15]. The process of cortical folding, or gyrification, continues well after term birth. Across most individuals, the gyrification is similar on a macroscopic level for the major structures, but differs on the level of the smaller sulci, even for monozygotic twins [2]. Before 20 weeks postmenstrual age (PMA) their brains' appearance have a high degree of similarity, but then progressively develop an individual character [25]. In this work, we try to model the subject specific development of the preterm and term neonatal brain, based on the dHCP dataset (see Section 3), containing brain MR images of neonates between 26 and 45 weeks PMA. This dataset poses challenges due to limited data availability, with each subject having scans from at most two or three different points in time, and the majority having only a single scan available.

Using an implicit neural representation (INR), we want to predict the healthy brain development from a given scan at a given PMA. That is, we want to predict an image of the same subject at a later or earlier point in time. Therefore, we have to find an age-agnostic representation of the identity of the subject, meaning we need to disentangle the age from the identity to be able to represent the same subject at different points in time. Modelling the development of the healthy brain could then serve as a basis for further downstream tasks, e.g. detection of abnormal brain development. INRs work by representing images (and other signals) as networks that take the coordinates of a pixel as input, and return the corresponding intensities. This is in contrast to convolutional neural networks (CNNs) or transformers, which process an entire image, i.e., a grid of pixels, at once. INRs have been successfully applied to the representation and modelling shapes [3,13], images [21], and recently also natural 3D scene reconstruction [20]. In the medical domain, they have been used for various tasks, such as image segmentation [22], shape completion [1], tomographic reconstruction [20,5], and image registration [23].

Contribution We show that an INR can be trained to model the neonatal brain development based on sparsely- and highly irregularly sampled data with respect to the time axis. Figure 1 provides an overview of the model. To enable the disentanglement along the time axis, we propose the following two methods that can be applied independently during training:

- A method to disentangle the subject's PMA at the time of the scan from its identity by enforcing a subject-specific latent space (SSL).
- An augmentation method with a global latent vector in the latent space. We call it stochastic global latent augmentation (SGLA). This performs similarly to SSL but is intended to make better use of subjects with only a single scan in the dataset.

We show how SSL and SGLA can improve the disentanglement, and with that, the predictions. Furthermore, we show that our INR approach can be run on hardware with limited GPU memory.

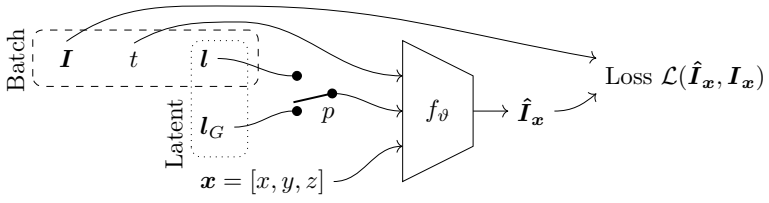


Fig. 1: Overview: The input to the network f_ϑ consists of the spatial coordinates $\boldsymbol{x}$ of the desired output pixel $\hat{\boldsymbol{I}}_{\boldsymbol{x}}$, the desired PMA $t \in T$, and a latent vector $\boldsymbol{l} \in \Lambda$, encoding the subject identity. The switch with probability p between $\boldsymbol{l}$ and the global latent vector $\boldsymbol{l}_G$ represents the SGLA.

2 Method

This work presents an approach based on an INR. This means that we model our images $\boldsymbol{I} \in \mathcal{I} = \{I : \Omega \to [0,1]\}$ at different points of time $t \in T$ as a function $f_\vartheta : \Omega \times T \times \Lambda \to [0,1]$ parametrized by ϑ, where $\Omega \subset \mathbb{R}^d$ is the spatial image domain, $T = [0,1]$ is the time domain (i.e. normalized PMA), and $\Lambda \subset \mathbb{R}^\lambda$ is our latent space. For any image $\boldsymbol{I} \in \mathcal{I}$ we denote the intensity at $\boldsymbol{x} \in \Omega$ as $\boldsymbol{I}_{\boldsymbol{x}}$. In the forward pass, for each pixel, the normalized input coordinates $\boldsymbol{x} \in \Omega$ are concatenated with the latent vector $\boldsymbol{l}$ and the corresponding PMA t and then passed to the network. Therefore, during training, for some given image $\boldsymbol{I}$ with corresponding PMA t and latent vector $\boldsymbol{l}$, and some input coordinates $\boldsymbol{x}$, we predict $\hat{\boldsymbol{I}}_{\boldsymbol{x}} := f_\vartheta(\boldsymbol{x}, t, \boldsymbol{l})$, and we optimize the ℓ^2-loss $\mathcal{L}(\hat{\boldsymbol{I}}_{\boldsymbol{x}}, \boldsymbol{I}_{\boldsymbol{x}}) = \|\hat{\boldsymbol{I}}_{\boldsymbol{x}} - \boldsymbol{I}_{\boldsymbol{x}}\|_2^2$ with respect to ϑ and $\boldsymbol{l}$. This is described in more detail in Section 2.2. For inference, we can project an input image into the latent space by optimizing the latent vector with respect to the reconstructed image. To then predict images $\hat{\boldsymbol{I}}^i$ for different points of time t_i, for a given latent vector $\boldsymbol{l}$, we can sample $\hat{\boldsymbol{I}}^i_{\boldsymbol{x}} := f_\vartheta(\boldsymbol{x}, t_i, \boldsymbol{l})$ over the whole image domain $\boldsymbol{x} \in \Omega$ for every t_i. This is described in detail in Section 2.2. The architecture of the network f_ϑ is described in Supplementary Material A.

2.1 Age Disentanglement in the Latent Space

In our setup, the latent space represents the identity of a subject. We want to disentangle the identity of the subject from its age. To this end, we propose the following two methods, which can be employed independently of each other:

Subject Specific Latent Vectors (SSL) Previously developed INRs for similar tasks use one latent vector per input image [1,22,3,13]. This is sufficient if we want our network to model a specific input image. In our case, we want the model to generalize the representation of a specific subject across the time domain T and to encode the subject's identity independently, i.e., disentangle the identity from the age. Given the availability of images from multiple time points for one subject, we propose using the *same* latent vector for all images of the *same* subject, with the goal of forcing the model to use only the age inputs $t \in T$ to encode the age of the subject and use the latent vector for encoding the identity of the subject only.

Stochastic Global Latent Augmentation (SGLA) The second method we propose for disentangling the subjects' age from their identity is based on a specific type of data augmentation during training: We introduce an additional *global* latent vector $\boldsymbol{l}_G$ for training. In each iteration of the stochastic gradient descent, the latent vector of the current batch is replaced with this global latent vector $\boldsymbol{l}_G$ with some probability p. It is intended to mimic SSL for those subjects that have only a *single* scan in the dataset: Similarly as in SSL, the network is trained to predict scans at different developmental stages for the same latent vector. Instead of a subject specific latent vector, we use the global vector $\boldsymbol{l}_G$ instead. This forces the network to take the age input t into account. We set the probability $p = 10\%$ to make it comparable to SSL, as about 10% of the subjects in the training set have more than one scan. We have performed an ablation with respect to p in Supplementary Material D.1.

2.2 Training & Inference

The network, along with the latent vectors of the training set, is jointly optimized using the *AdamW* optimizer [11] with a learning rate of $\mathrm{lr} = 10^{-4}$ and an ℓ_2-loss. Since the INRs can be processed pixel by pixel, we do not necessarily need to use the whole image for each training step. In our model, we can adjust the percentage of pixels that we randomly sample from a given image in each training step by a hyperparameter. We chose to use 5% for our experiments to fit the training of the 3D INRs on a GPU with 12GB memory. For simplicity, we used identical settings for the 2D case. This lets us easily adjust for the available memory size of a given GPU. The details of the pixel sampling are described in Supplementary Material D.2

If we want to use the entire image (be it for training or inference), we can still profit from the pixel-wise evaluations: We can serialize the whole input into micro-batches that are processed sequentially and accumulate the gradients with a constant amount of memory. We use this for the inference of the 3D images. The amount of memory used is then mainly dictated by the micro-batch size. These methods allow us to train models on devices with very little memory. We discuss the resource consumption in Section 3.

Inference During inference, we use the same loss as during training, but we keep the network fixed and only optimize the latent vector $\boldsymbol{l}$. Therefore, for a given image $\boldsymbol{I}^1$ with age t_1 as input, we optimize a latent vector $\boldsymbol{l}$ to minimize the difference between reconstruction $\hat{\boldsymbol{I}}^1_{\boldsymbol{x}} = f_\vartheta(\boldsymbol{x}, t_1, \boldsymbol{l})$ and the image $\boldsymbol{I}^1_{\boldsymbol{x}}$ over all $\boldsymbol{x} \in \Omega$, i.e.

$$\hat{\boldsymbol{l}} := \underset{\boldsymbol{l} \in \Lambda}{\operatorname{argmin}} \sum_{\boldsymbol{x} \in \Omega} \| f_\vartheta(\boldsymbol{x}, t_1, \boldsymbol{l}) - \boldsymbol{I}^1_{\boldsymbol{x}} \|_2^2. \tag{1}$$

We then use this vector $\hat{\boldsymbol{l}} \in \Lambda$ to generate a prediction $\hat{\boldsymbol{I}}$ for some desired age $t_2 \in T$ by computing $\hat{\boldsymbol{I}}_{\boldsymbol{x}} = f_\vartheta(\boldsymbol{x}, t_2, \hat{\boldsymbol{l}})$. The latent vectors are initialized with zeros and then optimized over 1000 step for the 3D case and 2000 for the 2D case, using *AdamW* with a learning rate of $\mathrm{lr} = 10^{-3}$.

2.3 Baseline: Denoising Diffusion Models

As a baseline, we use a denoising diffusion model with gradient guidance (DDM+GG) [26] to predict an image for a certain PMA, given some image of a different PMA. This method has been implemented for 2D images and has been shown to perform well on similar tasks such transforming portraits into younger or older versions, and simulating tumor growth over time.

DDM + GG uses two networks in tandem: the denoising and the regression network. The age conditioning is performed using the gradients of a regression network during the denoising process, which is trained to predict the subject's PMA. The denoising and the regression networks were trained to convergence (1M and 150k iterations, respectively) with $T = 1000$ noising and denoising steps. The number of noising and denoising steps for inference was $L = 600$ and the gradient scale $c = 3 \cdot 10^5$. We optimized both parameters L and c on the test set to make a fair comparison. Since this method was proposed for 2D images, we also trained our INR on the same 2D data (axial slices). The details are reported in Supplementary Material B.

3 Experiments & Results

In the following section, we report the results of our experiments. If not otherwise noted, the setup is as follows: From each subject in the test set, we consider two scans that were made at a different point of time, i.e., at a different PMA of the subject. As explained in Section 2.2, for a given subject, we first determine the latent vector $\hat{\boldsymbol{l}}$ based on the input image $\boldsymbol{I}^1$ and PMA t_1, that is, we optimize $\hat{\boldsymbol{l}}$ as defined in Equation (1). We then use this latent vector $\hat{\boldsymbol{l}}$ to generate a prediction $\hat{\boldsymbol{I}}^2$ for the PMA t_2 of the second scan $\boldsymbol{I}^2$ by computing $\hat{\boldsymbol{I}}^2_{\boldsymbol{x}} := f_\vartheta(\boldsymbol{x}, t_2, \hat{\boldsymbol{l}})$ for every $x \in \Omega$. We then compare the predicted image $\hat{\boldsymbol{I}}$ with the second scan $\boldsymbol{I}^2$, which serves as the ground truth for all metrics that we will introduce below. This allows us to quantify how well our model predicts the development process of the brain. To justify our contributions, we perform an ablation of our proposed method, that is, we conduct our experiments with and without SGLA and SSL, respectively, in 2D as well as in 3D.

Dataset We used the dHCP (third data release) dataset for our experiments [12,2]. It contains $T1$- and $T2$-weighted neonatal MR-scans of 329 subjects. We use the $T2$-weighted scans as these are preferred for assessing brain structure in fetal and neonatal MRI [12], due to the immature myelination. The postmenstrual age (PMA, in weeks) is available for each scan, ranging from 26 to 45 weeks with a median of 40.57. Our preprocessing is described in Supplementary Material C.

Ablation of SSL and SGLA and comparison to the Baseline To compare our predictions on the test set with the corresponding ground truths, we compute the peak-signal-noise-ratio (PSNR), the structural similarity index (SSIM) and the mean absolute error (MAE). We report the same metrics for the baseline DDM+GG in Table 1. Our proposed 2D INR with SGLA and SSL outperforms DDM+GG with respect to every metric. Furthermore, the performance decreases without SGLA or SSL (or neither). However, it should be noted that in the 2D case we only consider a slice of the 3D volume. Much of the anatomical context is therefore missing, making it more difficult to predict changes that are influenced by tissue not shown on the slice in question. For this purpose, we train our INR model on the 3D volumes as a whole and perform the same ablation of SGLA and SSL again. The results are reported in Table 1. The difference between the different models in terms of PSNR, SSIM and MAE is relatively small in absolute terms, because the image background is black, which is easy to predict for a model. In other words, if the background remains constant and only the foreground improves, this reduces the effect on these scores. However, we can still see that both SSL and SGLA improve all three scores. Furthermore, even in the presence of a number of subjects with multiple scans available, SGLA can be used in conjunction with SSL and still improve the performance, in 2D as well as in 3D.

In addition to the three metrics we use in the 2D case, in the 3D case, we can compute the head circumference (HC) based on our predictions (details in Supplementary Material E). Along with several other measurements, the HC is one of the most important factors in determining prenatal development [10]. Therefore, we used it to measure how well our proposed method performs concerning the disentanglement of the subjects' age. We compare the measured HC of our predictions

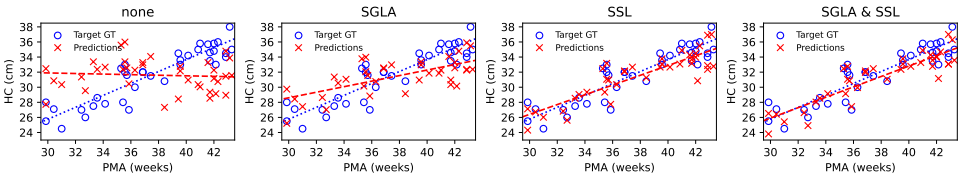
with the HC reported in the dHCP dataset, which we consider the ground truth. Table 1 displays the standard deviation σ (in cm) of the error between the measured HC of our prediction and the ground truth. Furthermore, we report the correlation coefficient r between the predicted HC and the ground truth HC. Notably, we can see that with neither SGLA nor SSL, the HCs of the predictions are almost uncorrelated with the ground truth HCs. Again, our proposed INR with SGLA and SSL outperforms the other ablated models in each of these measures or performs at least as well.

Table 1: Average scores over the test set for the 2D INRs and the 2D DDM+GG baseline, as well as for the 3D INRs.

			2D			3D				
Model	SGLA	SSL	PSNR	SSIM	MAE	PSNR	SSIM	MAE	HC σ	HC r
INR	n	n	19.5±2.6	0.642±0.058	0.0701±0.0178	22.9±3.1	0.804±0.036	0.0352±0.0091	3.850	0.173
	y	n	20.6±2.5	0.683±0.065	0.0571±0.0146	23.8±2.8	0.832±0.049	0.0274±0.0083	2.137	0.835
	n	y	21.3±2.1	0.715±0.075	0.0489±0.0124	24.5±2.4	0.850±0.054	0.0232±0.0079	1.071	0.962
	y	y	**21.4**±2.0	**0.722**±0.075	**0.0468**±0.0129	**24.6**±2.4	**0.853**±0.055	**0.0225**±0.0078	**0.964**	**0.968**
DDM+GG	-	-	19.3±2.1	0.632±0.088	0.0682±0.0158	-	-	-	-	-

To better understand these results, we plot the ground truth HC and the measured HC of our predictions in Figure 2 for all four INRs. We can see that the INR with neither SGLA nor SSL completely fails to capture the developmental process. Using either SGLA or SSL significantly improves the correlation between the HC of the prediction and the ground truth HC. Finally, using both jointly enables the model to capture the development process even better.

Fig. 2: Ablation of our model w.r.t. the HC. The blue circles show the HC ground truth, while the red x-es show the HC of our models' predictions on the test set. The slope of the linear regression of the target GT (blue dots) is 0.79. The slopes of linear regression of the predictions are none: -0.04, SGLA: 0.36, SSL: 0.65, SGLA & SSL: 0.69.



Qualitative Results In Figure 3, we show two examples from the test set. We show four images per subject: (1) the *input* $\boldsymbol{I}^1$ with its age t_1 that gets encoded into a latent vector, (2) the reconstruction $\hat{\boldsymbol{I}}^1$ of the input at t_1, (3) the *target ground truth* $\boldsymbol{I}^2$ from t_2 and (4) the prediction $\hat{\boldsymbol{I}}^2$ at age t_2. Note that for our model, t_2 does not necessarily have to be greater than t_1, we can also choose to look back in time to predict what a given brain has looked like in the past. The reconstructions display the input images with some loss of details due to the bottleneck created by the low-dimensional latent space. The predictions generally match the size and the contrast of the target ground

truth well, but have trouble predicting the exact shape of the cortical folds. This is, however, a difficult task, as even for monozygotic twins, the folds exhibit individual patterns [25]. On the bottom right, we see an interesting example where the mid-sagittal plane of the input is slightly off the vertical, in contrast to the target ground truth. Interestingly, however, the reconstruction, as well as the prediction, display this slight rotation, which means it must have been encoded in the latent vector.

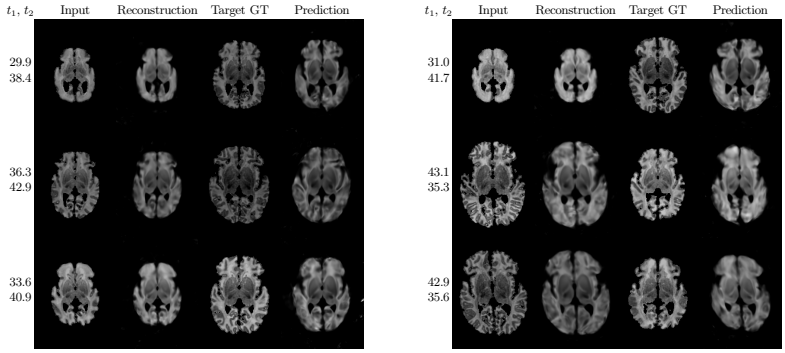


Fig. 3: Six 2D samples from the test set, along with the PMA t_1 of the input, and the PMA t_2 of the target ground truth image.

Predicted Average Development To gain insight into the development process that the trained network captured, we consider two methods of extracting an "average" brain for each age, similar to creating an age-resolved atlas. Firstly, since we have designed the latent space to represent the characteristics of a subject, it can be expected that averaging the latent vectors of a population (i.e., our training set) would result in an average-looking brain. Secondly, because we initialize the latent vectors with a zero-vector, it is also possible to use it for the generation of an "average" brain. Thus, we use these two (*zero-* and *average-*) latent vectors to generate a temporal sequence of images of the trained model with SSL and SGLA. Furthermore, we compare the two approaches with the IMAGINE fetal atlas [8], as well as the neonatal atlas of Schuh et al. [19]. The latter is also based on the dHCP dataset. In Figure 4, we report the measured HCs along with percentiles of a fetal-infant preterm growth chart [6,7]. For instance, for diagnosing microcephaly, bounds like the 1^{st} or 3^{rd} percentile or three standard deviations of the HC are being used [4]. Under these criteria, all four growth curves are within the normal range. Notably, the HC lines of our generated "average"-brains agree very well with the atlas of [19], which is based on the same dataset. The IMAGINE atlas [8], exhibits slightly larger HCs, but is created using a set of *fetal* MR-images, with a significantly smaller sample size of 81 scans. However, it also remains within the normal interval. The quality of the two generated sequences using the zero- and average latent vectoris different: The series generated by the average latent vector is more detailed and has less blurry features.

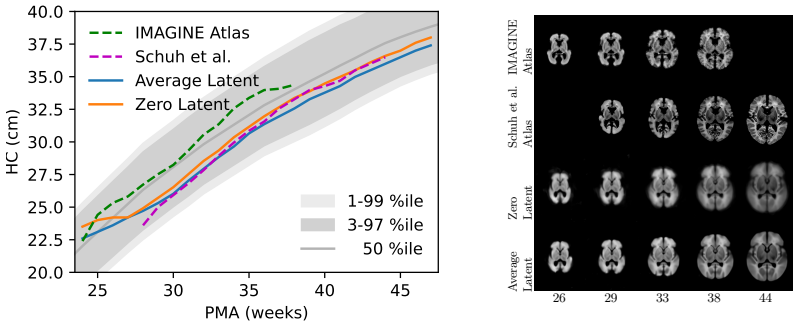


Fig. 4: Comparison of our "average" brain development with the IMAGINE atlas and the atlas of Schuh et al. in terms of HC growth curves Figure 4a, along with the quantiles reported in [6]. Furthermore, we show the corresponding axial slices in Figure 4b.

Computational Resources In Table 2, we report the time and GPU memory used. We used the same settings regarding pixel-sampling and micro-batching for the 2D- and 3D case. As elaborated in Section 2.2, we can reduce the number of pixels sampled per optimization step. Alternatively, we have the option to use micro-batching to trade memory for time. Finally, the number of parameters used in the 2D and 3D INRs is more than two orders of magnitude smaller than in the 2D DDM+GG baseline.

Table 2: Resources used for the INRs in 2D and 3D and the DDM+GG baseline.

	GPU Memory (in MB)		Time		
Model	Training	Inference	Training (total)	Inference (per Sample)	# Parameters
INR 3D	10350	10286	81.6 h	5.20 min	232 450
INR 2D	284	232	2.5 h	0.73 min	232 194
DDM 2D	7412	1332	76.8 h	1.80 min	113 669 762
Regression 2D	6412		7.9 h		86 786 817

4 Conclusion

We present a method for modelling the neonatal brain development using INRs. We show that it is necessary to disentangle the subject's identity from its age. We propose and evaluate two novel methods, SSL and SGLA. We implement them in 2D as well as in 3D. On the one hand, we compare the predictions with the ground truth with respect to image quality, and on the other hand also indirectly via the HC as an important development metric. We demonstrate how our two proposed solutions improve the results through the disentanglement. However, we note that the image synthesis with INRs still has open challenges such as the image quality, and

the modelling of processes like the cortical folding, which are additionally influenced by other factors than time. We anticipate that larger datasets would enable more expressive networks to be trained, which could in turn alleviate these issues to some degree. To avoid the need for coregistration of the scans as a preprocessing step, it would be interesting to extend the disentanglement also to geometric transformations. In future work, we would like to explore extending the model to predict segmentations, which could be used to generate actual atlases.

Acknowledgments. We are grateful for the support of the Novartis FreeNovation initiative and the Uniscientia Foundation (project #147-2018). We would also like to thank the NVIDIA Corporation for donating a GPU that was used for our experiments.

Data were provided by the developing Human Connectome Project, KCL-Imperial-Oxford Consortium funded by the European Research Council under the European Union Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. [319456]. We are grateful to the families who generously supported this trial.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Amiranashvili, T., Lüdke, D., Li, H.B., Menze, B., Zachow, S.: Learning shape reconstruction from sparse measurements with neural implicit functions. In: International Conference on Medical Imaging with Deep Learning. pp. 22–34. PMLR (2022)
2. Bozek, J., Makropoulos, A., Schuh, A., Fitzgibbon, S., Wright, R., Glasser, M.F., Coalson, T.S., O'Muircheartaigh, J., Hutter, J., Price, A.N., et al.: Construction of a neonatal cortical surface atlas using multimodal surface matching in the developing human connectome project. NeuroImage **179**, 11–29 (2018)
3. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019)
4. DeSilva, M., Munoz, F.M., Sell, E., Marshall, H., Kawai, A.T., Kachikis, A., Heath, P., Klein, N.P., Oleske, J.M., Jehan, F., et al.: Congenital microcephaly: case definition & guidelines for data collection, analysis, and presentation of safety data after maternal immunisation. Vaccine **35**(48Part A), 6472 (2017)
5. Fang, Y., Mei, L., Li, C., Liu, Y., Wang, W., Cui, Z., Shen, D.: Snaf: Sparse-view cbct reconstruction with neural attenuation fields. arXiv preprint arXiv:2211.17048 (2022)
6. Fenton, T.R.: A new growth chart for preterm babies: Babson and benda's chart updated with recent data and a new format. BMC pediatrics **3**, 1–10 (2003)
7. Fenton, T., Sauve, R.: Using the lms method to calculate z-scores for the fenton preterm infant growth chart. European journal of clinical nutrition **61**(12), 1380–1385 (2007)
8. Gholipour, A., Rollins, C.K., Velasco-Annis, C., Ouaalam, A., Akhondi-Asl, A., Afacan, O., Ortinau, C.M., Clancy, S., Limperopoulos, C., Yang, E., et al.: A normative spatiotemporal mri atlas of the fetal brain for automatic segmentation and analysis of early brain growth. Scientific reports **7**(1), 476 (2017)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

10. Kiserud, T., Piaggio, G., Carroli, G., Widmer, M., Carvalho, J., Neerup Jensen, L., Giordano, D., Cecatti, J.G., Abdel Aleem, H., Talegawkar, S.A., et al.: The world health organization fetal growth charts: a multinational longitudinal study of ultrasound biometric measurements and estimated fetal weight. PLoS medicine **14**(1), e1002220 (2017)
11. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
12. Makropoulos, A., Robinson, E.C., Schuh, A., Wright, R., Fitzgibbon, S., Bozek, J., Counsell, S.J., Steinweg, J., Vecchiato, K., Passerat-Palmbach, J., et al.: The developing human connectome project: A minimal processing pipeline for neonatal cortical surface reconstruction. Neuroimage **173**, 88–112 (2018)
13. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4460–4470 (2019)
14. Monteagudo, A., Timor-Tritsch, I.E.: Ultrasound of the fetal brain. Ultrasound Clinics **2**(2), 217–244 (2007)
15. Rubenstein, J., Rakic, P.: Patterning and cell type specification in the developing CNS and PNS: comprehensive developmental neuroscience, vol. 1. Academic Press (2013)
16. Salomon, L., Alfirevic, Z., Da Silva Costa, F., Deter, R., Figueras, F., Ghi, T.a., Glanc, P., Khalil, A., Lee, W., Napolitano, R., et al.: Isuog practice guidelines: ultrasound assessment of fetal biometry and growth. Ultrasound in obstetrics & gynecology **53**(6), 715–723 (2019)
17. Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., Baraniuk, R.G.: Wire: Wavelet implicit neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18507–18516 (2023)
18. Sarris, I., Ioannou, C., Chamberlain, P., Ohuma, E., Roseman, F., Hoch, L., Altman, D., Papageorghiou, A., Fetal, I., for the 21st Century (INTERGROWTH-21st), N.G.C.: Intra-and interobserver variability in fetal ultrasound measurements. Ultrasound in obstetrics & gynecology **39**(3), 266–273 (2012)
19. Schuh, A., Makropoulos, A., Robinson, E.C., Cordero-Grande, L., Hughes, E., Hutter, J., Price, A.N., Murgasova, M., Teixeira, R.P.A., Tusor, N., et al.: Unbiased construction of a temporally consistent morphological atlas of neonatal brain development. BioRxiv p. 251512 (2018)
20. Shen, L., Pauly, J., Xing, L.: Nerp: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. IEEE Transactions on Neural Networks and Learning Systems (2022)
21. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in neural information processing systems **33**, 7462–7473 (2020)
22. Stolt-Ansó, N., McGinnis, J., Pan, J., Hammernik, K., Rueckert, D.: Nisf: Neural implicit segmentation functions. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 734–744. Springer (2023)
23. Sun, S., Han, K., Kong, D., You, C., Xie, X.: Mirnf: Medical image registration via neural fields. arXiv preprint arXiv:2206.03111 (2022)
24. Tuzikov, A.V., Colliot, O., Bloch, I.: Brain symmetry plane computation in mr images using inertia axes and optimization. In: 2002 International Conference on Pattern Recognition. vol. 1, pp. 516–519. IEEE (2002)
25. White, T., Su, S., Schmidt, M., Kao, C.Y., Sapiro, G.: The development of gyrification in childhood and adolescence. Brain and cognition **72**(1), 36–45 (2010)
26. Wolleb, J., Sandkühler, R., Bieder, F., Cattin, P.C.: The swiss army knife for image-to-image translation: Multi-task diffusion models. arXiv preprint arXiv:2204.02641 (2022)

Chapter 9

Discussion and Conclusion

The goal of this work was to develop image analysis methods that can deal with the inherent size challenges of three-dimensional MRI and CT brain images.

As we elaborated in Section 4.2.6, GPUs are instrumental for the use of deep learning, especially for image processing. A crucial property of GPUs for training for deep learning models is the memory size. However, the memory size is directly linked to the computation power of GPUs and is associated with their cost, as discussed in Section 4.2.6.2. To enable a more widespread adoption of deep learning models, and also to promote research in lower-income countries, it is an important step to lower the barrier of entry.

9.1 Discussion

In the first project (see Chapter 6), we developed and validated an unsupervised anomaly detection and localization method. At the core of the method is a novel self-supervised training objective that is used as a surrogate task for anomaly detection. It consists of predicting the location of small patches within coregistered scans. When trained on images free of anomalies, the network implicitly learns a representation of the anatomy. In an inference step, the model predicts the patches' position and compares them to the actual positions from where the patch was sampled. The error is what we base the anomaly detection on. Many anomaly detection methods rely on a restoration approach (see Section 5.4). Many of them used datasets with anomalies that appear hyperintense, which was criticized [165], as there may be much more straightforward methods that can perform equally well.

We showed that our proposed model can detect and localize brain hemorrhages and fractures in head CT scans. While hemorrhages appear hyperintense, cranial fractures consist of small fracture lines or displacement of the fractured pieces. This shows that our method is not as much affected by the concerns mentioned above. However, the patch-based approach limits the spatial resolution: If we consider two patches with a great overlap, the error in the predicted positions will only change by a small amount, as the network is a continuous function. This is clearly visible in the examples of intracranial

hemorrhages. Around the border of the hyperdense regions, the error continuously increases. Similarly, if the anomalies have a small extent, this proposed method may fail to detect them. Also, if multiple anomalies of different sizes are present, just thresholding the error maps might overestimate the size of the larger ones and underestimate the size of the smaller ones. To address this limitation, we think that it could be attractive to extend the method into a multiscale approach.

We showed that the network used in the proposed approach can be chosen relatively small, without significantly sacrificing performance. The patch-based approach and the small network size allow the network to be trained on consumer-GPUs with 1 to 5 GB of memory, depending on the specific settings.

The second project we presented (see Chapter 7) built on a novel segmentation method based on denoising diffusion models [140]. This method uses denoising diffusion models to generate the segmentation through the denoising process, which is conditioned on the image we would like to segment. The inherent stochasticity of the denoising diffusion models defines a distribution of segmentation masks and an *implicit ensemble*. This allows us to quantify the uncertainty in these generated segmentation masks. The method was implemented for two-dimensional images. To make it more suitable for medical images, we aimed to implement it for 3D volumes.

Using a naive reimplementation of the same architecture for 3D data was impossible, as it would have consumed too many resources, including memory. We therefore explored ways to reduce the resource consumption. A few changes were architectural, i.e., we optimized the architecture of the network itself, but we also introduced the patch-wise training. The architecture we ended up with is purely convolutional and can, therefore, process images of various sizes. During training, we used patches of half the size of the original, which reduced the memory requirements to a manageable level. Simultaneously, as such a patch only contains an eighth of the original image, each optimization step also just consumes roughly one eighth of the number of operations, decreasing the time per step. As we also explored the accelerated sampling during the denoising process, we chose to use the DDIM sampling scheme instead of the DDPM, as a higher image quality was reported during the sampling, with very few sampling steps [142]. While the quality of the segmentations decreases with the accelerated sampling, we could also show how that can be traded with the ensemble size.

Our proposed model performs the segmentation task quite well, but does not outperform state-of-the-art methods. This was, however, also not our primary focus. As we used the DDIM sampling scheme, the only source of noise was the initial noise vector x_T. The experiments showed that there can be isolated misclassified voxels in the segmentation. We attribute this to extreme samples (i.e., in the tails of the initial $\mathcal{N}(0, I)$-distribution) that could not be corrected through the denoising process. It is plausible that using the DDPM sampling process could alleviate this issue.

While we could successfully bring these diffusion models to the 3D domain, our proposed approach also introduced more complexity into an already rather complex type of model. Alternative solutions that have tackled the issue of the resource consumption have since been proposed and include latent diffusion models [170], as well as wavelet

diffusion models [171].

The goal of the third project (see Chapter 8) was modeling the brain development in the neonatal period. The dataset included one MRI for most subjects. For a few subjects, it included and multiple MRIs from different points in time. We explored the use of implicit neural representations to train a prior that can be used to model the development process on a per-subject basis. A key problem was the disentanglement of the subject-specific development process (we called it the subjects' *identity*) from the age. We developed a method that enabled this disentanglement, which makes use of the few subjects that have multiple scans available. Based on similar principles, we developed a second method that makes use of the subjects, of which there is only a single scan in the dataset. We showed that combined, these methods greatly improve the disentanglement, which helps the INR to capture the process more accurately. A concurrently developed method was proposed to generate atlases that are not subject-specific to avoid the need for the disentanglement by regressing over the learned latent vectors of the training set based on the age [172].

We have shown that when processing high dimensional data (here we have three spatial dimensions as well as time), INR-based methods can be advantageous with respect to the computational resources. As individual pixels or voxels can be processed independently, we can split the evaluation of an INR into a set of these voxels (a *microbatch*) to fit the memory constraint that is imposed by the available hardware.

Our proposed model used gradient descent to directly optimize a latent vector to encode an image during inference. We adopted this encoder-less technique from [122, 121, 118]. However, it has been shown [119] that a separate convolutional encoder can be used as well. This could eliminate this optimization step during inference.

While the generative power of INRs has already been widely explored for shape representations [122, 119], it has not yet been explored as much for images. A challenge of current INRs for images with generative capability is their spectral bias, as discussed in Section 5.2.4. Our proposed model struggles to capture finer details. They most prominently include the formation of gyri and sulci. This process is challenging to model due to many influences that are not yet completely understood. It is also dependent on environmental factors [173], which can partially explain why a network can fail to capture its complexity. Furthermore, it is possible that the chosen architecture can be improved.

Specifically, using prior knowledge about the growth, [172] proposed a decomposed parametrization of "scale" and "shift" in sinusoidal activations functions to specifically encourage lower frequencies along the time dimension.

We focussed on a *pure* INR-based method. Other methods combine multiple techniques, like, e.g., in [174], where a GAN was proposed to produce a *triplane* representation that included an INR as a decoder, to recover 3D structure from multiple 2D views.

9.2 Conclusion

In this work, we presented three different applications of deep learning for processing brain MR and CT scans. We showed that while deep learning generally is a resource-hungry machine learning technique, it is possible to tackle certain problems with more limited resources. For MR and CT scan this is especially important, as they consist largely of three-dimensional, or four-dimensional data when we include time series, and therefore inherently have a large memory footprint to begin with. In this work, we proposed novel memory-efficient methods for anomaly detection and localization, brain-tumor segmentation via diffusion models and the modeling of early brain development. Deep learning continues to progress and enables things that might have been considered impossible even just a decade back. But currently, the trend seems to be making models bigger, also increases the energy consumption and limits their use to those who can afford the necessary hardware. With this work, we hope to raise awareness about the resource consumption of deep learning models and to encourage researchers to also investigate methods that are less resource-hungry, in order to make them more accessible.

Bibliography

[1] J. Ferrandis and A. Segal. "L'essor de la radiologie osseuse pendant la guerre de 1914-1918". In: *Rhumatologie Pratique* 266 (2009), pp. 48–50. URL: https://horizon14-18.eu/wa_files/l_27essor_20de_20la_20radiologie_20osseuse.pdf.

[2] W. G. Bradley. "History of Medical Imaging". In: *Proceedings of the American Philosophical Society* 152.3 (2008), pp. 349–361. ISSN: 0003049X. URL: http://www.jstor.org/stable/40541591 (visited on 08/27/2024).

[3] D. Wheeler and E. Spencer. "Simplified planigraphy". In: *Radiology* 34.4 (1940), pp. 499–502.

[4] N. England. *Diagnostic Imaging Dataset Annual Statistical Release 2022/23*. Tech. rep. NHS England, 2023. URL: https://www.england.nhs.uk/statistics/wp-content/uploads/sites/2/2023/11/Annual-Statistical-Release-2022-23-PDF-1.3MB-1.pdf.

[5] C. X. Y. Goh and F. C. H. Ho. "The growing problem of radiologist shortages: perspectives from Singapore". In: *Korean Journal of Radiology* 24.12 (2023), p. 1176.

[6] A. Al Kabbani, T. Walizai, J. Feger, et al. *Skull. Reference article, Radiopaedia.org*. https://radiopaedia.org/articles/skull. Accessed: 2024-07-02.

[7] The Database Center for Life Science. *BodyParts3D: A 3D Human Body Parts Dataset*. Accessed: 2024-08-15. 2011. URL: https://lifesciencedb.jp/bp3d/?lng=en.

[8] E.-J. Speckmann and W. Wittkowski. *Handbuch Anatomie: Bau und Funktion des menschlichen Körpers*. Elsevier, 2009.

[9] J. Jones, R. Sharma, B. Botz, et al. *Subarachnoid space. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-5857. Accessed: 2024-07-14.

[10] C. Hacking, T. Walizai, D. Bell, et al. *Cerebrospinal fluid. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-58893. Accessed: 2024-09-02.

[11] Frank Gaillard. *Layers of the scalp and meninges*. Accessed: 2024-08-14. 2010. URL: https://radiopaedia.org/cases/layers-of-the-scalp-and-meninges-illustrations.

[12] F. Gaillard, T. Foster, D. Bell, et al. *Neuron. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-51710. Accessed: 2024-09-05.

[13] H. Blumenfeld. *Neuroanatomy through Clinical Cases 2nd Ed.* Sinauer Associates Incorporated, 2009. ISBN: 978-0-87893-058-6.

[14] F. Gaillard, D. Bell, C. Hacking, et al. *Brain. Reference article, Radiopaedia.org.* https://radiopaedia.org/articles/glial-cells. Accessed: 2024-07-02.

[15] F. Gaillard, C. Hacking, and H. Knipe. *Astrocytes. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-51707. Accessed: 2024-07-14.

[16] F. Gaillard, C. Hacking, and A. Skandhan. *Oligodendrocytes. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-51712. Accessed: 2024-07-14.

[17] F. Gaillard, D. Bell, C. Hacking, et al. *Microglia. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-51717. Accessed: 2024-07-14.

[18] E. N. Marieb and K. N. Hoehn. *Human Anatomy & Physiology, Global Edition.* Pearson Education, 2018. ISBN: 978-1-292-26093-8.

[19] J. G. Betts et al. *Anatomy and Physiology.* Accessed: [Insert Access Date]. Houston, Texas: OpenStax, 2013. URL: https://openstax.org/books/anatomy-and-physiology/pages/1-introduction.

[20] W. A. Engle et al. "Age terminology during the perinatal period." In: *Pediatrics* 114.5 (2004), pp. 1362–1364.

[21] R. R. O'Rahilly and F. Müller. *The Embryonic Human Brain: An Atlas Of Developmental Stages.* John Wiley & Sons, 2006. ISBN: 978-0-471-97307-2.

[22] B. Cohen-Sacher et al. "Sonographic developmental milestones of the fetal cerebral cortex: a longitudinal study". In: *Ultrasound in Obstetrics and Gynecology: The Official Journal of the International Society of Ultrasound in Obstetrics and Gynecology* 27.5 (2006), pp. 494–502.

[23] A. Makropoulos et al. "The developing human connectome project: A minimal processing pipeline for neonatal cortical surface reconstruction". In: *Neuroimage* 173 (2018), pp. 88–112.

[24] J. Bozek et al. "Construction of a neonatal cortical surface atlas using multimodal surface matching in the developing human connectome project". In: *NeuroImage* 179 (2018), pp. 11–29.

[25] S. Bakas et al. "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features". In: *Scientific data* 4.1 (2017), pp. 1–13.

[26] F. Gaillard, R. Chieng, H. Knipe, et al. *Brain tumors. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-4986. Accessed: 2024-07-18.

[27] H. A. Wanis et al. "The incidence of major subtypes of primary brain tumors in adults in England 1995-2017". In: *Neuro-Oncology* 23.8 (2021), pp. 1371–1382.

[28] PDQ Adult Treatment Editorial Board. *PDQ Adult Central Nervous System Tumors Treatment*. Bethesda, MD: National Cancer Institute. Updated 2024-01-05. Available at: https://www.cancer.gov/types/brain/patient/adult-brain-treatment-pdq. [PMID: 26389458]. Accessed: 2024-07-22.

[29] F. Sciacca. *Organs at risk. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-80650. Accessed: 2024-07-21.

[30] H. Knipe, M. Elfeky, M. Saber, et al. *Traumatic brain injury. Reference article, Radiopaedia.org*. https://radiopaedia.org/articles/traumatic-brain-injury. Accessed: 2024-07-02.

[31] N. I. for Health and C. Excellence. *Head Injury: assessment and early management, guideline NG232*. https://www.nice.org.uk/guidance/ng232. Accessed: 2024-07-02.

[32] F. Gaillard. *Skull Fractures. Reference article, Radiopaedia.org*. (Visited on 07/04/2024).

[33] S. Chilamkurthy et al. "Deep learning algorithms for detection of critical findings in head CT scans: a retrospective study". In: *The Lancet* 392.10162 (2018), pp. 2388–2396.

[34] J. M. Gebel and J. P. Broderick. "Intracerebral hemorrhage". In: *Neurologic clinics* 18.2 (2000), pp. 419–438.

[35] S. Tenny and W. Thorell. "Intracranial Hemorrhage". eng. In: *StatPearls*. Treasure Island (FL): StatPearls Publishing, 2024. URL: http://www.ncbi.nlm.nih.gov/books/NBK470242/ (visited on 07/04/2024).

[36] F. Gaillard, R. Sharma, H. Knipe, et al. *Intracranial hemorrhage. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-1518. Accessed: 2024-07-21.

[37] *Vascular Neurology Board Review: An Essential Study Guide*. Springer International Publishing, 2020. ISBN: 978-3-030-52551-4 978-3-030-52552-1. (Visited on 07/04/2024).

[38] *Medical Imaging Systems: An Introductory Guide*. Vol. 11111. Lecture Notes in Computer Science. Springer International Publishing, 2018. ISBN: 978-3-319-96519-2 978-3-319-96520-8. (Visited on 07/26/2024).

[39] D. Weishaupt, V. D. Köchli, and B. Marincek. *Wie funktioniert MRI?: Eine Einführung in Physik und Funktionsweise der Magnetresonanzbildgebung*. Springer, 2014. ISBN: 978-3-642-41615-6 978-3-642-41616-3. (Visited on 07/04/2024).

[40] D. W. McRobbie et al. *MRI from Picture to Proton*. Third. Cambridge University Press, 2017. ISBN: 978-1-107-64323-9. (Visited on 07/26/2024).

[41] G. B. Chavhan et al. "Principles, techniques, and applications of T2*-based MR imaging and its special applications". In: *Radiographics* 29.5 (2009), pp. 1433–1449.

[42] M. Bernstein, K. King, and X. Zhou. *Handbook of MRI Pulse Sequences*. English (US). Publisher Copyright: © 2004 Elsevier Inc. All rights reserved. Elsevier Inc., Sept. 2004. ISBN: 9780120928613.

[43] F. Gaillard, Y. Baba, D. Bell, et al. *MRI sequences (overview). Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-37346. Accessed: 2024-09-20.

[44] M. Niknejad, R. Chieng, Y. Baba, et al. *Fluid attenuated inversion recovery. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-21760. Accessed: 2024-08-26.

[45] Y. Weerakkody, A. Murphy, Y. Baba, et al. *MRI artifacts. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-16585. Accessed: 2024-08-13.

[46] D. Bell, T. Walizai, Y. Mellam, et al. *Ionizing radiation. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-59177. Accessed: 2024-08-06.

[47] J. Jones, A. Campos, L. McKay, et al. *Stochastic effects. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-5099. Accessed: 2024-09-02.

[48] A. Gerstmair, C. Worsley, A. Murphy, et al. *As low as reasonably achievable (ALARA). Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-35183. Accessed: 2024-08-11.

[49] P. Mudgal, A. Campos, R. Chieng, et al. *X-ray production. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-25428. Accessed: 2024-08-06.

[50] Raymond Chieng. *Energy spectrum of tungsten anode at 80kVp and 100 kVp.* Accessed: 2024-08-12. 2023. URL: https://radiopaedia.org/cases/energy-spectrum-of-tungsten-anode-at-80kvp-and-100-kvp?lang=us.

[51] M. Nadrljanski, L. McKay, C. Worsley, et al. *Attenuation coefficient. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-9035. Accessed: 2024-08-07.

[52] M. Nadrljanski, A. Campos, R. Chieng, et al. *Computed tomography. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-9027. Accessed: 2024-08-06.

[53] B. Botz, A. Campos, J. Feger, et al. *Photon-counting computed tomography. Reference article, Radiopaedia.org.* https://doi.org/10.53347/rID-78900. Accessed: 2024-08-07.

[54] User Obynel, Wikimedia Commons. *Installation of a CT Scan.* Accessed: 2024-08-07. 2020. URL: https://commons.wikimedia.org/wiki/File:Installation_of_a_CT_Scan.jpg.

[55] *Computed Tomography: Approaches, Applications, and Operations.* Springer International Publishing, 2020. ISBN: 978-3-030-26956-2 978-3-030-26957-9. (Visited on 08/08/2024).

[56] E. Ahishakiye et al. "A survey on deep learning in medical image reconstruction". In: *Intelligent Medicine* 1.03 (2021), pp. 118–127.

[57] T. D. DenOtter and J. Schubert. "Hounsfield unit". In: *StatPearls.* StatPearls Publishing, 2019. URL: https://www.ncbi.nlm.nih.gov/books/NBK547721/ (visited on 08/08/2024).

[58] A. Murphy, A. Campos, M. Niknejad, et al. *Beam hardening. Reference article, Radiopaedia.org*. https://doi.org/10.53347/rID-48590. Accessed: 2024-08-25.

[59] P. R. Patel and O. De Jesus. "CT Scan". In: *StatPearls*. Treasure Island (FL): StatPearls Publishing, 2023. URL: https://www.ncbi.nlm.nih.gov/books/NBK567796/ (visited on 08/09/2024).

[60] D. P. Frush. "Radiation, thoracic imaging, and children: radiation safety". In: *Radiologic Clinics* 49.5 (2011), pp. 1053–1069.

[61] A. Krizhevsky, G. Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[62] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[63] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[64] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[65] S. J. Prince. *Understanding Deep Learning*. Accessed: 2024-07-30. The MIT Press, 2023. URL: http://udlbook.com.

[66] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[67] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[68] J. L. Ba, J. R. Kiros, and G. E. Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[69] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[70] Y. Wu and K. He. "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[71] K. Yamaguchi et al. "A neural network for speaker-independent isolated word recognition." In: *ICSLP*. 1990, pp. 1077–1080.

[72] A. Vaswani. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[73] A. Van Den Oord, O. Vinyals, et al. "Neural discrete representation learning". In: *Advances in neural information processing systems* 30 (2017).

[74] K. He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[75] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[76] A. Paszke et al. *PyTorch Documentation: Autograd*. https://pytorch.org/docs/stable/notes/autograd.html. Accessed: 2024-08-20. 2024.

[77] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[78] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[79] K. He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[80] B. Widrow and M. A. Lehr. "30 years of adaptive neural networks: perceptron, madaline, and backpropagation". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1415–1442.

[81] A. G. Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Journal of machine learning research* 18.153 (2018), pp. 1–43.

[82] N. Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. Version 1.0. Accessed: 2024-08-15. 2007. URL: https://developer.download.nvidia.com/compute/cuda/1.0/NVIDIA_CUDA_Programming_Guide_1.0.pdf.

[83] N. Corporation. *cuDNN: NVIDIA CUDA Deep Neural Network Library*. Accessed: 2023-10-01. 2023. URL: https://developer.nvidia.com/cudnn.

[84] M. Abadi et al. "{TensorFlow}: a system for {Large-Scale} machine learning". In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 2016, pp. 265–283.

[85] A. Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[86] J. de Vries. *Learn OpenGL*. Kendall & Welling, 2020. URL: https://learnopengl.com/Getting-started/Transformations.

[87] N. Corporation. *Convolutional Layers User's Guide*. https://docs.nvidia.com/deeplearning/performance/dl-performance-convolutional/index.html. Accessed: 2024-07-30. 2023.

[88] N. Corporation. *Matrix Multiplication Background User's Guide*. https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html. Accessed: 2024-07-30. 2023.

[89] T. M. Aamodt, W. W. L. Fung, and T. G. Rogers. *General-Purpose Graphics Processor Architecture*. Synthesis Lectures on Computer Architecture. Series Editor: Margaret Martonosi, Princeton University. Morgan & Claypool: Morgan & Claypool, 2018. URL: https://store.morganclaypool.com.

[90] R. Krashinsky et al. *NVIDIA Ampere Architecture In-Depth*. Accessed: 2024-08-27. May 2020. URL: https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/.

[91] PyTorch Foundation. *PyTorch Documentation: Data*. Accessed: 2024-08-27. 2023. URL: https://pytorch.org/docs/stable/data.html.

[92] F. Bieder et al. "Memory-Efficient 3D Denoising Diffusion Models for Medical Image Processing". In: *Medical Imaging with Deep Learning*. Vol. 227. Proceedings of Machine Learning Research. PMLR, July 2024, pp. 552–567.

[93] PyTorch Foundation. *PyTorch Documentation: Checkpoint*. Accessed: 2024-09-06. 2023. URL: https://pytorch.org/docs/stable/checkpoint.html.

[94] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[95] Wikipedia. *Weak supervision — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Weak%20supervision&oldid=1230360937. Accessed: 2024-08-26. 2024.

[96] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[97] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[98] M. Wiatrak, S. V. Albrecht, and A. Nystrom. "Stabilizing generative adversarial networks: A survey". In: *arXiv preprint arXiv:1910.00927* (2019).

[99] J. Xu et al. "Diversity-promoting GAN: A cross-entropy based generative adversarial network for diversified text generation". In: *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2018, pp. 3940–3949.

[100] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[101] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel recurrent neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1747–1756.

[102] J. Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[103] T. Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[104] H. Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[105] K. He et al. "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009.

[106] A. Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[107] Z. Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[108] R. Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[109] J. Ho, A. Jain, and P. Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[110] J. Song, C. Meng, and S. Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[111] V. Sitzmann et al. "Implicit neural representations with periodic activation functions". In: *Advances in neural information processing systems* 33 (2020), pp. 7462–7473.

[112] B. Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[113] V. Saragadam et al. "Wire: Wavelet implicit neural representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 18507–18516.

[114] Y. Fang et al. "Snaf: Sparse-view cbct reconstruction with neural attenuation fields". In: *arXiv preprint arXiv:2211.17048* (2022).

[115] A. Kaseorg. *Paint Starry Night, objectively, in 1kB of code*. Code Golf & Coding Challenges StackExchange. Accessed: 2024-08-05. URL: https://codegolf.stackexchange.com/a/149835/.

[116] M. Garnelo et al. "Conditional neural processes". In: *International conference on machine learning*. PMLR. 2018, pp. 1704–1713.

[117] M. Bemana et al. "X-fields: Implicit neural view-, light-and time-image interpolation". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15.

[118] T. Amiranashvili et al. "Learning shape reconstruction from sparse measurements with neural implicit functions". In: *International Conference on Medical Imaging with Deep Learning*. PMLR. 2022, pp. 22–34.

[119] L. Mescheder et al. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[120] L. Shen, J. Pauly, and L. Xing. "NeRP: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.1 (2022), pp. 770–782.

[121] N. Stolt-Ansó et al. "Nisf: Neural implicit segmentation functions". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, pp. 734–744.

[122] J. J. Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[123] Z. Chen et al. "Multiresolution deep implicit functions for 3d shape representation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13087–13096.

[124] J. Chibane, G. Pons-Moll, et al. "Neural unsigned distance fields for implicit function learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21638–21652.

[125] J. McGinnis et al. "Single-subject multi-contrast MRI super-resolution via implicit neural representations". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, pp. 173–183.

[126] N. Rahaman et al. "On the spectral bias of neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 5301–5310.

[127] B. B. Damodaran et al. "Improved Positional Encoding for Implicit Neural Representation based Compact Data Representation". In: *arXiv preprint arXiv:2311.06059* (2023).

[128] T. Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM transactions on graphics (TOG)* 41.4 (2022), pp. 1–15.

[129] Z. Liu et al. "FINER: Flexible spectral-bias tuning in Implicit NEural Representation by Variable-periodic Activation Functions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 2713–2722.

[130] Y. Chen et al. "HOIN: High-Order Implicit Neural Representations". In: *arXiv preprint arXiv:2404.14674* (2024).

[131] S. Gao et al. "Implicit diffusion models for continuous super-resolution". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 10021–10030.

[132] Z. Erkoç et al. "Hyperdiffusion: Generating implicit neural fields with weight-space diffusion". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 14300–14310.

[133] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny. "Adversarial generation of continuous images". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 10753–10764.

[134] S. Peng et al. "Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9054–9063.

[135] J. Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[136] T. Chen, C. Wang, and H. Shan. "Berdiff: Conditional bernoulli diffusion model for medical image segmentation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2023, pp. 491–501.

[137] J. Wolleb et al. "Binary noise for binary tasks: Masked bernoulli diffusion for unsupervised anomaly detection". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2024, pp. 135–145.

[138] A. Bansal et al. "Cold diffusion: Inverting arbitrary image transforms without noise". In: *Advances in Neural Information Processing Systems* 36 (2024).

[139] C. Saharia et al. "Palette: Image-to-image diffusion models". In: *ACM SIGGRAPH 2022 conference proceedings*. 2022, pp. 1–10.

[140] J. Wolleb et al. "Diffusion models for implicit image segmentation ensembles". In: *International Conference on Medical Imaging with Deep Learning*. PMLR. 2022, pp. 1336–1348.

[141] T. Amit et al. "Segdiff: Image segmentation with diffusion probabilistic models". In: *arXiv preprint arXiv:2112.00390* (2021).

[142] A. Q. Nichol and P. Dhariwal. "Improved denoising diffusion probabilistic models". In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.

[143] J. Wolleb et al. "The swiss army knife for image-to-image translation: Multi-task diffusion models". In: *arXiv preprint arXiv:2204.02641* (2022).

[144] X. Liu et al. "More control for free! image synthesis with semantic diffusion guidance". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 289–299.

[145] J. Wyatt et al. "Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 650–656.

[146] G. Couairon et al. "Diffedit: Diffusion-based semantic image editing with mask guidance". In: *arXiv preprint arXiv:2210.11427* (2022).

[147] F. Isensee et al. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation". In: *Nature methods* 18.2 (2021), pp. 203–211.

[148] J. Ma. "Cutting-edge 3D medical image segmentation methods in 2020: Are happy families all alike?" In: *arXiv preprint arXiv:2101.00232* (2021).

[149] S. Andermatt, S. Pezold, and P. Cattin. "Multi-dimensional gated recurrent units for the segmentation of biomedical 3D-data". In: *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer. 2016, pp. 142–151.

[150] P. Moeskops et al. "Adversarial training and dilated convolutions for brain MRI segmentation". In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. Springer. 2017, pp. 56–64.

[151] B. H. Menze et al. "The multimodal brain tumor image segmentation benchmark (BRATS)". In: *IEEE transactions on medical imaging* 34.10 (2014), pp. 1993–2024.

[152] R. Azad et al. "Loss functions in the era of semantic segmentation: A survey and outlook". In: *arXiv preprint arXiv:2312.05391* (2023).

[153] L. Maier-Hein et al. "Metrics reloaded: recommendations for image analysis validation". In: *Nature methods* 21.2 (2024), pp. 195–212.

[154] D. Karimi and S. E. Salcudean. "Reducing the hausdorff distance in medical image segmentation with convolutional neural networks". In: *IEEE Transactions on medical imaging* 39.2 (2019), pp. 499–513.

[155] C. Baur et al. "Deep autoencoding models for unsupervised anomaly segmentation in brain MR images". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. Springer. 2019, pp. 161–169.

[156] X. Chen and E. Konukoglu. "Unsupervised detection of lesions in brain MRI using constrained adversarial auto-encoders". In: *arXiv preprint arXiv:1806.04972* (2018).

[157] M. Heer et al. "The ood blind spot of unsupervised anomaly detection". In: *Medical Imaging with Deep Learning*. PMLR. 2021, pp. 286–300.

[158] D. Zimmerer et al. "Unsupervised anomaly localization using variational auto-encoders". In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV 22*. Springer. 2019, pp. 289–297.

[159] C. Baur et al. "Autoencoders for unsupervised anomaly segmentation in brain MR images: a comparative study". In: *Medical Image Analysis* 69 (2021), p. 101952.

[160] S. Andermatt et al. "Pathology segmentation using distributional differences to images of healthy origin". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. Springer. 2019, pp. 228–238.

[161] J.-Y. Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[162] J. Wolleb, R. Sandkühler, and P. C. Cattin. "DeScarGAN: Disease-Specific Anomaly Detection with Weak Supervision". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV*. Springer-Verlag, 2020, pp. 14–24. ISBN: 978-3-030-59718-4.

[163] T. Schlegl et al. "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks". In: *Medical image analysis* 54 (2019), pp. 30–44.

[164] J. Wolleb et al. "Diffusion models for medical anomaly detection". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2022, pp. 35–45.

[165] F. Meissen, G. Kaissis, and D. Rueckert. "Challenging current semi-supervised anomaly segmentation methods for brain MRI". In: *International MICCAI brainlesion workshop*. Springer. 2021, pp. 63–74.

[166] D. Zimmerer et al. "MOOD 2020: A public Benchmark for Out-of-Distribution Detection and Localization on medical Images". In: *IEEE Transactions on Medical Imaging* 41.10 (2022), pp. 2728–2738.

[167] S. N. Marimont and G. Tarroni. *Achieving state-of-the-art performance in the Medical Outof-Distribution (MOOD) challenge using plausible synthetic anomalies*. 2023.

[168] M. Baugh et al. "Many tasks make light work: Learning to localise medical anomalies from multiple synthetic tasks". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, pp. 162–172.

[169] F. Bieder et al. "Position Regression for Unsupervised Anomaly Detection". In: *International Conference on Medical Imaging with Deep Learning*. PMLR. 2022, pp. 160–172.

[170] J. Kim and H. Park. "Adaptive latent diffusion model for 3d medical image to image translation: Multi-modal magnetic resonance imaging study". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 7604–7613.

[171] P. Friedrich et al. "Wdm: 3d wavelet diffusion models for high-resolution medical image synthesis". In: *MICCAI Workshop on Deep Generative Models*. Springer. 2024, pp. 11–21.

[172] M. Dannecker et al. “CINA: Conditional Implicit Neural Atlas for Spatio-Temporal Representation of Fetal Brains”. In: *arXiv preprint arXiv:2403.08550* (2024).

[173] T. White et al. “The development of gyrification in childhood and adolescence”. In: *Brain and cognition* 72.1 (2010), pp. 36–45.

[174] E. R. Chan et al. “Efficient geometry-aware 3d generative adversarial networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16123–16133.