The Out-of-Distribution Blindspot of Unsupervised Lesion Detection using Variational Autoencoders

Master's Thesis

Matthäus Heer
Department of Information Technology and Electrical Engineering, ETH Zurich
Department of Physics, University of Zurich

Advisors: Dr. Shadi Albarqouni, Xiaoran Chen, Janis Postels
Supervisor: Prof. Dr. Ender Konukoglu, Prof. Dr. Jan Unkelbach

February 25, 2021

Abstract

MRI (Magnetic Resonance Imaging) scans pose the primary screening method to detect, assess and segment brain pathologies amongst other diseases for diagnosis and subsequent treatment planning. These checks are often performed by radiologists who examine such abnormalities manually. Since this is a time-consuming process, there is a need for more effective and potentially more robust solutions for this task. Due to recent successes on many visual tasks such as image classification or segmentation, the machine learning community has undertaken large efforts to assist in the analysis of such images or even automate this lengthy process entirely.

Recently, generative models, such as Variational Autoencoders, have been successfully deployed to perform unsupervised lesion detection as a promising alternative to state-of-the-art supervised approaches. These unprecedented models are trained on healthy patients data exclusively and measure abnormality of test samples by assessing their deviation to the learned healthy in-distribution training data. While current methods solve the problem to some extent, they might be prone to errors in real-world scenarios since the current assumption for the factors of a sample being Out-of-Distribution (OOD) does not include the notion of distributional shifts due to, for example, varying scanning protocols and parameters. This renders these methods vulnerable to actual OOD samples for which reason this work aims to understand and disentangle those underlying root causes for a sample to be predicted to be OOD.

The main contributions of this work are as follows. First, it reveals that the hypothesis of partial blindness or entanglement stated above is of great concern when applying such models in practice and is typically overlooked thus far. Thereto, we propose measures to disentangle the influencing factors for a sample to be predicted anomalous or not. Recent developments in the field of OOD detection in the form of the *WAIC* and *DoSE* scores are being explored and applied to the case of unsupervised anomaly detection. Next, a proposal of weak prior knowledge about the nature of lesions in comparison with samples which have undergone a domain-shift is introduced via a normalized entropy score. Furthermore, an approach following the notion of learning disentangled representations via strong latent space regularization is being assessed for usability in the task of disentangling these underlying factors.

Results show that the proposed methods are unable to improve the currently employed metrics for unsupervised lesion detection or successfully disentangle the underlying factors for a test sample to be OOD. However, since it could be shown that entangled factors are a major concern in these frameworks, further research to enable safe deployment is strongly encouraged. Last but not least, several insights into the encoding principles and training mechanisms of unsupervised generative models have been made and documented to facilitate future research.

Acknowledgements

Tremendous thank goes to my supervisors, **Dr. Shadi Albarqouni**, **Xiaoran Chen** and **Janis Postels**. Our weekly meetings gave me food for thought and were an essential part of my journey carrying out this thesis. Their unique competencies in a variety of fields made it possible to have fruitful, productive and fun discussions.

Furthermore, I want to thank **Prof. Dr. Jan Unkelbach** for his guidance throughout my studies and **Prof. Dr. Ender Konukoglu** for letting me complete my thesis in his research group at the Computer Vision Lab at ETH Zürich.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Medical Motivation

Accumulations of diseased cells within the brain and medulla spinalis are referred to as tumors of the central nervous system. Celebral tumors (tumors in the brain) only make up for around two percent of all malignancies (cancerous tumors) and can thus be regarded as relatively rare occurrences. Nonetheless, around 500-700 people fall ill every year due to celebral tumors in Switzerland alone [krebsliga.ch, 2020], in most cases being being affected by a glioma. The World Health Organization (WHO) differentiates more than 100 types of celebral tumors [Arevalo et al., 2017] from which the gliomata are found to be the most frequently occurring ones. Gliomata are then further classified into grades of severity ranging from grade I to IV (Glioblastomas), where the latter make up for around 50% of all glioma. [Wolff et al., 2010] reported 5 year survival rates ranging from around 10 to 19% for children, depending on the chemotherapeutic treatment. Survival rates for adults are reported to be even lower and strongly depends on the treatment possibilities ranging from 50% to 0%. It stands to reason that a reliable tumor detection system is of utmost importance. Although the methods presented in this work are very general and applicable to an unlimited range of imaging datasets in which one might be interested to detect anomalies, they are first and foremost tested on MRI brain imaging data with the aim to detect and segment such malignant tissue.

Symptoms indicating malignant brain tumors include strong headaches, visual disorders, confusedness, dissociative convulsions or unconsciousness amongst others [Aigner and Stephens, 2016]. On an initial check-up, a doctor will conduct an anamnesis followed by a physical and neurological examination to test the workings of the patients central nervous system. This is typically done with various tests targeting the patients responsiveness and coordination skills and observing reactions of specific muscle groups to external stimuli. In case the results substantiate concerns regarding a potential cerebral tumor, imaging procedures such as MRI (Magnetic Resonance Imaging (cf. Sec. 2.8), CT (Computer Tomography) and PET (Positron Emission Tomography) are being employed next. MRI scans pose the primary screening method to detect, assess and segment brain pathologies for diagnosis and subsequent treatment planning. In practice, a doctor might be interested in the existence and extend of a tumor mass which he would abstract from the sectional images of a MRI scan which might also be registered with other imaging modalities, such as PET. Since this process is not only time-consuming (a single scan might consist of several hundred slices) but also error-prone, large efforts have been undertaken by the machine learning community to automate the detection and segmentation of lesions from brain MRI scans.

Similar to other fields into which Deep Learning has found its way into, such as Autonomous Driving, the medical domain can be regarded as a field with high-consequence decision making. Although machine learning based diagnostics hold great potential to assist and befriend medical

practitioners, there are open questions as to how safe and robust such systems are to date. This work investigates the applicability and trustability of Deep Learning based anomaly detection frameworks, specifically so-called unsupervised lesion detection approaches.

1.2 Focus of this Work

While supervised Deep Learning approaches (cf. Sec. 2.1) , often based on the infamous U-Net architecture [Ronneberger et al., 2015], contribute state-of-the-art lesion segmentation techniques [Isensee et al., 2020], they are constrained to the distribution of anomalies used during training and the need for corresponding pixel-wise ground-truth labels from domain experts. This process of crafting pixel-level lesion annotations is expensive and subject to an inherent inter- and intra-rater ambiguity [Moraal et al., 2010]. Unsupervised methods on the other hand have the potential to serve as a class-agnostic anomaly detection framework and might act as quality assurance tool for doctors and practitioners without the need to curate specialized training datasets.

Nevertheless, unsupervised anomaly detection techniques exploit OOD detection as their working principle, making them vulnerable to actual OOD samples during inference. Those are samples that might or might not contain lesions but more importantly originate from a different domain compared to the training distribution, which might be due to different scanner models used or changed parameters such as magnetic field strength during scanning. While the issue of unreliable model behavior for OOD samples is well-known and has received much attention in the case of supervised approaches in the past, also in the medical field [Mårtensson et al., 2020], it is even more pronounced for unsupervised anomaly detection, which equates anomaly detection with OOD detection, by nature. Since models performing unsupervised anomaly detection in the field of MRI will be presented with data whose generating process is governed by numerous influencing factors that might provoke a domain-shift, this is of great concern for safe model deployment and is usually not addressed assuming that test samples follow the same distribution like the training data. Furthermore, datasets that contain both healthy and lesional samples are typically not publicly available which complicates the investigation of the severity of model performance degradation under such domain-shifts and makes this a particularly challenging problem since there is no access to lesional samples originating from the same domain. Thus, it is important to investigate whether common approaches can disentangle the underlying factors for a sample to be OOD - i.e. whether it is lesional or not.

This work explores the effects of domain-shifted test data in the setting of unsupervised lesion detection of brain tumors, which has not been addressed before. More explicitly, we asses commonly deployed lesion detection metrics as well as recently developed OOD detection techniques in the field of generative modelling. We show that domain-shift effects are usually entangled with effects the actual lesions have on the prediction. Thus, it should be desirable to have metrics or techniques which tackle either one or the other.

1.3 Thesis Organization

The remainder of this work is organized as follows. Chapter 2 will give an overview on basic machine learning principles and provide the theoretical background and foundations for the specific techniques, concepts and evaluation procedures used throughout this work. Chapter 3 will give an overview on the most relevant works in the field of generative modelling, unsupervised lesion detection and OOD detection that constitute essential building blocks for this work. Chapter 4 presents our approaches and methodologies. Chapter 5 will present experiments and corresponding results along with visual examples while chapter 6 will present the discussion and conclusion of our experiments and give some suggestions for future work. The appendix A holds additional

information on implementation, further examples and a copy of the paper submission which is currently under review for the 2021 MIDL (Medical Imaging with Deep Learning) conference, titled *The OOD Blindspot of Unsupervised Lesion Detection*.

Chapter 2

Fundamentals

This chapter provides a brief summary of Machine and Deep Learning fundamentals on which the remainder of this work relies on. It starts by discussing core concepts of learning paradigms, the historic background of (deep) neural networks, their components and training mechanism. Finally, it introduces core concepts from probability and information theory. The reader is expected to be familiar with simple concepts from probability theory such as random variables or conditional probability distributions.

2.1 Supervised vs. Unsupervised Learning

2.1.1 Supervised Learning

Suppose to be given a dataset consisting of a set of samples $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with corresponding labels $\mathbf{y} = \{y^{(i)}\}_{i=1}^{N}$. A label $y^{(i)}$ could either be categorical or continuous. Supervised learning aims to infer a direct mapping $f(\mathbf{x}) \mapsto y$ for any sample $\mathbf{x}$ with corresponding label y, given some training data $\mathbf{X}$. This is achieved by training a model which acts as a generic function approximator f. Depending on the approach taken, those models are often regarded as black box function approximators, meaning that, while they achieve to learn the actual mapping f, it is hard to determine the reasoning of a network to come up with a particular prediction. While this is true to a large extent for most models used in Deep Learning approaches in the vision domain (cf. Sec. 2.4), classical machine learning models, such as decision trees, might offer more insights into the decision making procedure. In case y is continuous, this process is denoted as a regression task while categorical labels result in classification tasks (cf. Fig. 2.1 (b)), that is, finding a decision boundary in a potentially high dimensional feature space to separate unseen instances from different classes. Note that the labels y could be rather complex, for example, bounding boxes for the task of object detection within an an image or pixel-level label maps for the task of semantic segmentation, that is, assigning class labels to every pixel within an input image.

2.1.2 Unsupervised Learning

In the unsupervised setting, there are no available labels y associated with data points $\mathbf{x}$ during the learning phase. Thus, the goal is to learn some underlying, hidden structure from the data. Common tasks include clustering, dimensionality reduction, representation learning or density estimation (cf. Fig. 2.1 (a), (c), (d) & (e)). Note that unsupervised anomaly detection, introduced in Sec. 3.3, heavily relies on the principle of feature or representation learning based on some training data distribution, which in this works case is usually a set of healthy MRI brain scans.

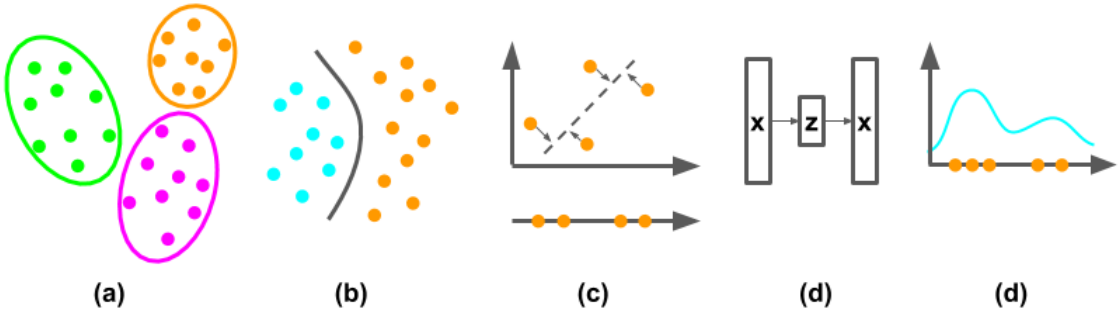


Figure 2.1: Examples from supervised and unsupervised machine learning. (a) unsupervised clustering in a 2-dimensional feature space with three classes, (b) learning a binary decision boundary in supervised classification within a 2-dimensional feature space, (c) dimensionality reduction from two to one dimension using Principal Component Analysis (PCA) by projecting data points onto e lower-dimensional manifold that represents the largest amount of variance in the projected data, (d) representation (feature) learning in an autoencoder setting by encoding a sample x into a latent representation z before decoding it back into an input space representation, (e) density estimation for one-dimensional datapoints using Kernel Density Estimation (KDE).

Furthermore, Variational Autoencoders [Kingma and Welling, 2014], introduced in Sec. 3.1.2, implicitly perform density estimation, that is, they learn the data distribution of the training data at hand. While the number of systems relying on unsupervised learning, that is, learning about the world around us without explicit supervision, in the industry lack far behind the more obvious use cases of supervised approaches, they are believed to constitute an essential ingredient to pave the road towards what is referred to as general artificial intelligence. It should be noted that, while supervised approaches might learn highly efficient representations for some task at hand, signals contained in the data which might be very relevant for other tasks could be ignored completely since they are of no use for the specific task the model has been trained for. An ideal unsupervised model on the other hand would learn information rich, transferable representations of data which should capture all there is to learn about the data and thus truly understand the underlying distribution.

2.2 Discriminative vs. Generative Models

2.2.1 Discriminative Models

Discriminative models do model the probability $p(y|\mathbf{x})$ for some data point $(\mathbf{x}, y)$. Intuitively speaking, they do not model densities of some data distributions but only have to learn a conditional distribution shown above. For example, in a supervised classification task, this would account to concentrating efforts on learning the decision boundary between classes, often making them superior over generative models which tackle a much broader objective.

2.2.2 Generative Models

Generative approaches model the joint distribution $p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$, that is they have to estimate the underlying density distribution which is a challenging task in most cases. The generative process can then be described by first sampling from $p(y)$ and then, given a particular y, generate a sample $p(\mathbf{x}|y)$. In the generative setting, one is usually concerned with generating new samples similar to those of a training data distribution $p_{data}(\mathbf{x})$, which a generative model approximates

by learning a distribution $p_{model}(\mathbf{x})$. Thus, generative models tackle the problem of density estimation, either implicitly, like Generative Adversarial Networks (GAN) [Goodfellow et al., 2014], or explicitly, like Variational Auto Encoders (cf. Sec. 3.1.2) which approximate the true density (cf. Sec. 3.1.2). While it is a flattering objective to be able to reproduce samples following a certain data distribution, the true power of generative models lies in their ability to learn a model of how the samples from the training data distribution are actually being created and to be able to do so they have to *understand* this distribution. For images, the most successful generative models follow the GAN framework, which is able to generate high fidelity, close to realistic looking samples. The VAE framework based on variational inference on the other side is known to produce blurry images. Nevertheless, it has the attractive property of approximating what is known as the posterior distribution explicitly by means of a latent space distribution and thus exposes more insights into the generative process and the underlying data distribution. It offers the possibility to sample explicitly from this latent space to generate specific types or classes of data samples. More details will be given in chapter 3, where those frameworks are discussed in great detail.

2.3 Representation Learning

Representation learning is concerned with learning useful representations of data making it easier to extract information for downstream tasks such as classification or regression [Bengio et al., 2012]. More generally, this includes extracting and potentially disentangling the underlying hidden features of data which are relevant for the task at hand. Prior the the Deep Learning era, much attention has been put into engineering feature extraction pipelines to boost the models predictive performances which was not only labour intensive but limited the general applicability of artificial intelligence in the first place due to engineered bias and cognitive limitations. Representation learning is particularly interesting in the field of unsupervised learning where, due to the absence of labelled data point, the only hope is to learn data representations to draw conclusions from. Multi-task learning models, that is, models that perform multiple predictions given a single data sample, can benefit from Representation Learning by sharing representations across multiple predictive tasks, for example in the form of a shared backbone network. Finally, learned representations in a supervised or unsupervised fashion can be used by models for specific tasks by fine-tuning the representation using only little data. This process is known as transfer learning and is widely adopted in the field of image classification. The models used throughout this work (cf. Sec. 3.1) reside in the field of representation learning by learning a compressed representation of an input image to tackle the problem of unsupervised lesion detection (cf. Sec. 3.3). Fig. 2.2 depicts in an attractive way, how convolutional neural networks (cf. Sec. 2.4.6) build up a visual vocabulary by combining low level features such as edges and textures to high level representations which resemble actual real-life objects.

2.4 Neural Networks

2.4.1 History

The first approach to model computation in the human brain dates back to 1943, when W. McCulloch and W. Pitts attempted to model the functionality of so-called neurons (cf. Sec. 2.4.2) [McCulloch and Pitts, 1988], the structural and functional primary unit of the nervous system. In the course of this, they modeled a simple neural network by means of electrical circuits. A major breakthrough, at least conceptually, in training such networks dates back to 1986 when [Rumelhart et al., 1986] introduced and popularized the so-called back propagation algorithm (cf. Sec. 2.4.5) to the field of neural networks. Deep learning, a sub-field of machine learning, refers to the train-

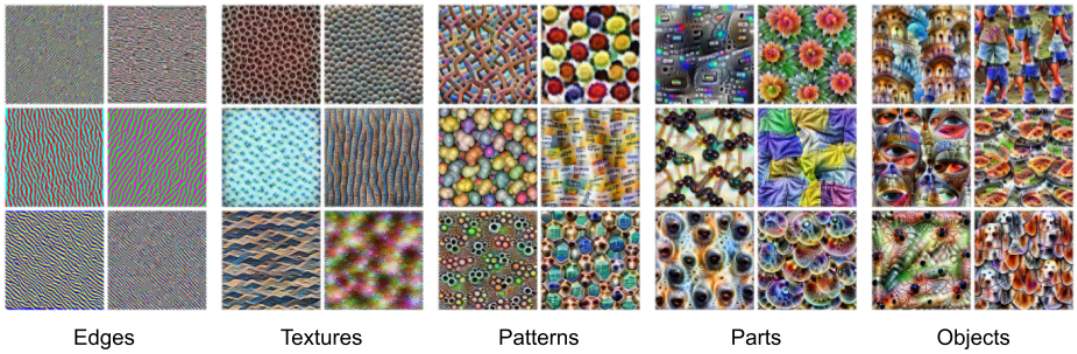


Figure 2.2: Visualization of images which have been generated through an optimization procedure that activate particular neurons (cf. Sec. 2.4.2) in a given layer maximally for the GoogLeNet [Szegedy et al., 2014] architecture. Those feature images can be regarded as "visual words" and the response of a particular neuron to any query image indicates whether a particular feature is present or not within the image. Going from left to right, the depth of layers increases which indicates that primitives, such as simple edges, encoded in early layers, are being used to construct more high level representations in deeper layers of the network, that is, closer to the output.
Source: [Olah et al., 2017].

ing of high capacity neural networks consisting of multiple layers of hidden neurons. Although the mathematical foundations to train such networks has been developed decades ago, interest has risen dramatically in the recent years due to successes in their deployment in the fields of Computer Vision [Krizhevsky et al., 2012] and Natural Language Processing. Higher availability of large scale datasets, which are necessary to train such large networks, as well as strong increases in compute power, first and foremost the possibility to use Graphical Processing Units (GPUs) for training, are regarded as the main drivers for the triumphal march of deep learning in the last decade.

2.4.2 Neurons and Neural Networks

Neural networks draw inspiration from biological neurons in the brain and spinal cord and the signal passing along their connections. A neuron or nerve cell represents the main functional unit of the nervous system. It reacts to stimuli of sensing organs and forwards this information to the brain or spinal cord, receives motor signals from the brain to perform, e.g. muscle contractions, or simply connects to other neurons for message parsing. It possesses one so-called axon filament at which ending there is a synapse. Signals are passed through an axon and the synapse with an electrochemical pulse, the so-called action potential, which is launched when the neuron gets excited electrically.

Fig. 2.3 depicts the basic computations happening on the level of a single neuron within an artificial neural network which can be understood as a simplified model of a biological processing unit of our brains. The neuron has a number of input signals $x_1, x_2, ..., x_n$, where each signal x_i is associated with a weight w_i. The input signals are being combined as a weighted sum

$$z = b + \sum_{i=1}^{n} x_i * w_i \tag{2.1}$$

where b is the so-called bias. Subsequently, the intermediate result z is forwarded through an

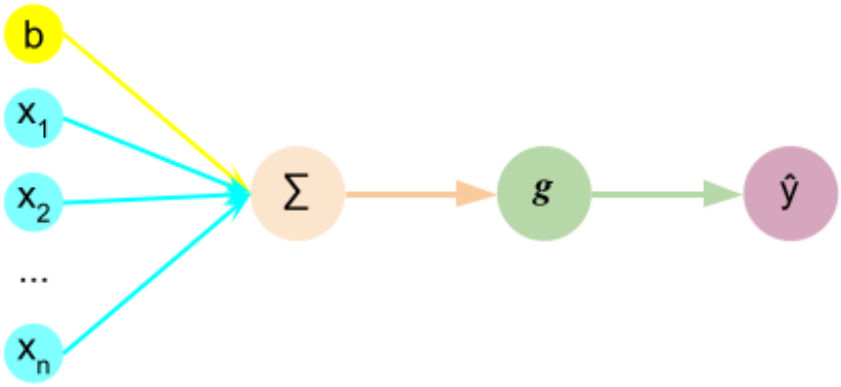


Figure 2.3: The basic computational layout of a neuron within a neural network. A weighted sum $\sum$ of input signals along the bias is passed through a non-linearity g to produce a neuron output value $\hat{y}$.

activation function (cf. Sec. 2.4.3) g to obtain the output $\hat{y}$ of the neuron

$$y = g(z) \tag{2.2}$$

Stacking layers of neurons where each activation value of some neuron residing in one particular layer acts as an input value of a subsequent layer yields what is referred to as a multi-layer perceptron (cf. Fig. 2.4), a basic neural network architecture which is often referred to as being the ”vanilla”, i.e. default, neural network in textbooks. It is made up of several so-called fully-connected layers (cf. Sec. 2.4.6) of neurons while the output value of one neuron is computed by Equ. 2.2. Other types of layers will be introduced in Sec. 2.4.6. A *forward pass* refers to the computation carried out when calculating neuron activation values consecutively starting with the input layer, passing the hidden layers and finally arriving at the output layer which represents the networks prediction.

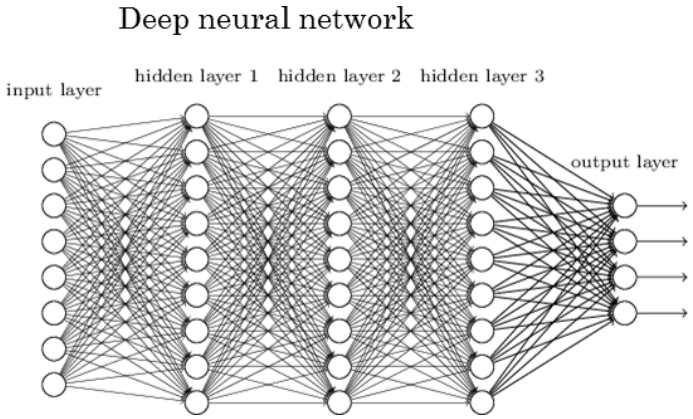


Figure 2.4: A multilayer perceptron with an eight-dimensional input layer, three hidden layers and a four-dimensional output layer.
Source: https://www.rsipvision.com/

2.4.3 Activation Functions

Without passing the weighted sum z through a non-linear activation function g, a neural network would essentially become a linear regressor, incapable of modelling complex, non-linear, high

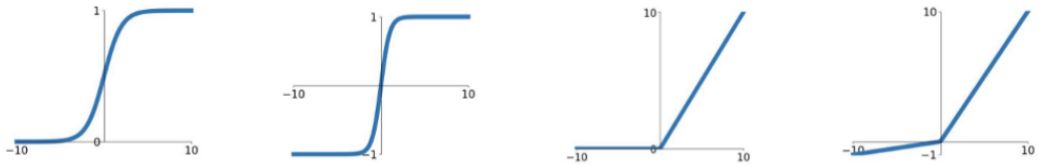


Figure 2.5: A selection of the most popular activation functions used. From left to right: Sigmoid, Tanh, ReLU (Recitified Linear Unit) and leaky ReLU activation functions.

dimensional mappings. Non-linear activation functions allow for the network to learn high order polynomials beyond one degree. The activation function decides on the amount of activation for a particular neuron while a non-zero bias shifts the activation function in either direction.

The most commonly used activation functions [Nwankpa et al., 2018] are depicted in Fig. 2.5. Historically, a common choice for an activation has been the *Sigmoid*, also known as the logistic function

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

However, it suffers from what is referred to as the *gradient vanishing problem*, that is, due to the near-zero derivative at the activation functions tails, the gradients vanish during the backpropagation algorithm (cf. Sec.2.4.5) towards earlier layers in the network. In consequence, earlier layers are not being updated, or learn as much, as the later layers and the network might get stuck in a (very sub-optimal) local minimum. However, since the Sigmoid function values resides in the interval $[0, 1]$, it is a valid choice for the output layer in the setting of binary classification. There, it might be desirable to predict a class-membership probability-distribution.

$$g(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.4}$$

More recently, the *ReLU* (Rectified Linear Unit)

$$g(z) = max(0, z) \tag{2.5}$$

and *leaky ReLU*

$$g(z) = \begin{cases} z & \text{if z greater 0} \\ az & \text{else} \end{cases} \tag{2.6}$$

have become popular, where a is commonly set to 0.01. ReLU-based activation functions guarantee faster computation since no exponentials or divisions have to be performed. Maybe more importantly, these functions have derivatives different from zero for large range of input values which speeds up training and counteracts the vanishing gradient problem.

Interestingly, although new state-of-the-art activation functions might outperform older activation functions or variants thereof, new network architectures usually rely on tested versions, such as the ReLU [Nwankpa et al., 2018]. A very common pattern seems to be to use ReLU activations for hidden layers while Softmax activations are used on output layers in multi-class classification tasks to turn the so-called logits, the neuron outputs before passing them through an activation function, into a normalized probability distribution of class memberships. The Softmax function is defined as

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \; k = 1, ..., K \tag{2.7}$$

for K classes and is a generalisation of the logistic function seen above. The function transforms the components of a k-dimensional input vector z with real components into a k-dimensional output vector $\sigma(z)$ whose components are squished into the range $(0, 1]$ and sum up to 1.

2.4.4 The Loss Function

Training Neural Networks can be regarded as solving a non-linear optimization problem. Which is, tuning the networks internal parameters such that for a given input, a desired output is being produced. Hence, it is required to define an objective function, which in this context is referred to as a loss function. By performing backward passes (cf. Sec. 2.4.5) subsequent to forward passes through the network, its weights and biases are adjusted such that the occurring loss is being minimized.
A natural choice for such an objective is the *mean squared error* or l2-loss which is popular in the field of optimization and reads

$$L_{MSE} = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{2.8}$$

where N is the number of samples while $\hat{y}_i$ refers to the prediction for some input sample $\mathbf{x}_i$ and y_i is the corresponding ground truth label. When applying this loss function to images, the loop in Equ. 2.8 could run over all pixels in an image or all the pixels of a batch of images as explained in the subsequent section. Note that in practice, often the *sum of absolute differences* or l1-loss is used

$$L_1 = \frac{1}{N}\sum_{i=0}^{N}|y_i - \hat{y}_i| \tag{2.9}$$

In classification tasks, it is common practice to put a Softmax activation on the network's final layer for multi-class classification and a Sigmoid for binary classification. Subsequently, the *cross-entropy* loss is being computed as

$$L_{CE} = -\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) \tag{2.10}$$

resulting in *categorical* and *binary cross-entropy* losses respectively. Sec. 2.4.5 will describe how the network weights are being updated with respect to the sample- or batch-wise loss such that a forward pass through the network produces an output closer to the desired prediction during training.

2.4.5 Training

Training a neural network refers to the process of successively updating its parameters (weights and biases) such that the network yields correct predictions for unseen examples for some task at hand. For example, a classification network should output the label "cat" when trained to recognise images of various animals and "cat" is a valid category in the trainind dataset. In that case, a test image is being fed into the input layer, a forward pass with the corresponding calculations (cf. Equ. 2.2) takes place and the neuron associated with the "cat"-category should yield the highest value of all neurons in the output layer, denoting a high probability for the sample to be of type "cat". Updating the weights correctly such that this outcome is going to be likely is achieved with the so-called backpropagation algorithm discussed next.

Optimization and The Backpropagation Algorithm

The training procedure is an interplay between forward passes of samples or batches of samples through the network and backward passes (which is essentially done via the backpropagation algorithm) to adjust the weights in a direction which minimizes the objective function value for

the training batch passed to the network. Let C denote the total cost of the network, that is the average loss per sample for all samples present in the training data. The direction in which the weights and biases, represented by the vector $\mathbf{w}$, should be changed is then given by the negative gradient of this total cost function

$$-\nabla C(\mathbf{w}) \tag{2.11}$$

Note that referring to the term *weight* often includes the biases and addresses all of the network's parameters. The i-th entry of this gradient vector represents the sensitivity of the cost function C to changes of the weight w_i. The actual magnitude of the update depends on the learning rate η, which is a training hyper-parameter (cf. Sec. 2.4.5) and is usually set by the operator and potentially adjusted over the course of training. The weight update for one particular training step, that is, one forward pass of (a batch of) data and the corresponding backward pass, is given by

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla C(\mathbf{w}) \tag{2.12}$$

The question remains on how to compute this gradient vector for such a complex function like a neural network with millions of parameters. Sec. 2.4 introduced the idea of *forward passes*, that is, calculating the output of the network given some input. The backpropagation algorithm is concerned with *backward passes* to compute the gradient of the neural networks cost function with respect to every single parameter. In other words, if we know the influence of every parameter to the overall loss, we know how to turn the knobs and dials to adjust the parameters in order to increase the final loss. This works by starting at the very last layer and applying the chain rule consecutively moving backwards through the network until the input layer is reached.

Stochastic and Mini-batch Gradient Descent

In practice, it is computationally very demanding or infeasible to compute the influence to the gradient of the cost function of every single training example in every single training step. Instead, the training set is shuffled and random samples or batches of samples are being drawn for which the gradient is then being computed to update the weights accordingly. The former strategy is denoted as *stochastic gradient descent* (a single random sample is responsible for a full update of the networks weights) while the latter is called *mini-batch gradient descent* (a set of training sample - a so-called mini-batch yields one update step of the networks weights). While those strategies likely do not yield the true gradient of the cost function at any time, they work well in practice and let the network approach a local minimum much faster than computing the gradient including all training samples in every step which would embody the actual *gradient descent* algorithm. Fig. 2.6 compares the behaviour of the three variants of gradient descent schematically. Note that basic gradient descent yields the optimal update of parameters but is also computationally very demanding. On the other hand, stochastic gradient descent produces a very wiggly path in the cost function's space with a very cheap update of weights. Mini-batch gradient descent combines the best of both worlds with descent parameter updates and feasible computational efforts and is thus done in practice. The algorithm is written-out in Listing 1.

It should be noted that cost functions of high-dimensional networks, that is, networks with a high number of parameters, are usually far from being convex like the example shown in Fig. 2.6. Thus, training such a network likely results in getting stuck in a local minimum. While this might sound problematic at first, practice has shown that those local minima yield "good enough" predictive power for the networks to be deployed in a variety practical applications.

Hyper-parameters

We have learned before that neural networks act as automatic feature extractors. That is, they learn representations or features of data without the need for human experts to curate or extract features

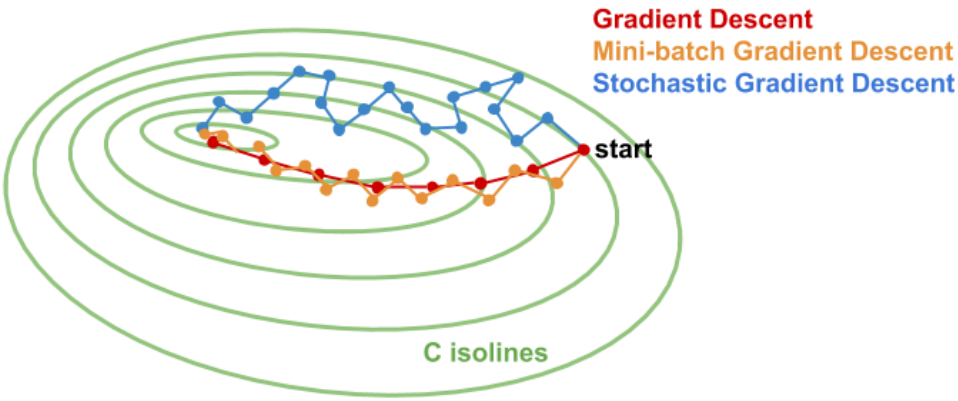


Figure 2.6: Schematic comparison of *stochastic gradient descent*, *mini-batch gradient descent* and standard *gradient descent* for a hypothetical two-dimensional cost function represented by its isolines. While the gradient descent algorithm uses the true gradient calculated by incorporating all data at hand, stochastic or mini-batch gradient descent might eventually converge to the same or qualitatively equal local minimum with much less computational resources required.

Algorithm 1: Mini-batch Gradient Descent

0) initialize network parameters randomly;
while *cost decreases* **do**
    1) sample a mini-batch from the training set;
    2) forward pass the batch through the network to obtain predictions;
    3) calculate the mean gradient of the cost function for the mini-batch;
    4) update the networks paramters accordingly;
end

from samples using prior knowledge. However, to train neural networks, one has to set certain parameters that cannot be (this is actually subject to research since it is clearly advantageous to learn as many aspects involved in a neural network training procedure as possible) learned by the model and are commonly referred to as hyper-parameters.

A number of hyper-parameters are hidden in the actual **model architecture**. This might include the number of layers, the width, height and number of convolutional filters. Also the type of activation functions can be regarded as part of the hyperparameters.

In Sec. 2.4.5 we have already come across the **learning rate** η. It is usually set to some small value like 0.0001 in the applications discussed in this work. However, there exist adaptive schemes where the learning rate is decreased over time as the network approaches some minimum of its cost function. If the learning rate is too large, the weight updates will be large as well and the trajectory will overshoot minima which might drive the loss into oscillation or divergence and prohobit convergence. On the other hand, if the learning rate is too small training will take significantly longer.

The mini-batch gradient descent algorithm introduced earlier relies on the sampling of a number of samples from the training data. The size of those batches is denoted as the **batch size** and mainly depends on the available memory of the underlying compute hardware. Since larger batch

mean more parallelization, training is usually faster and one usually chooses whatever fits into the memory of the GPU at hand. That being said, the choice of batch size can influence the training dynamics.

2.4.6 Common Components and Techniques

This section gives an overview over common components and techniques used in modern deep learning. It is not exhaustive and focuses on the most important and relevant works in the field, discussing techniques which are being used in the models deployed in experiments throughout this work.

Layered Structure

Artificial neural networks are usually composed of various numbers of neuronal layers. The output of the computation belonging to a neuron in a particular layer, as defined in Equ. 2.2, is being forwarded as part of an input vector to a neuron in a subsequent layer. Over the last decades, the number of layers has been seen continuous growth. While the early LeNet5 network [Lecun et al., 1998] was composed of only five layers, recent networks such as the Resnet architecture [He et al., 2015] contain hundreds of layers. The term *Deep Learning* refers to training networks with multiple numbers of (hidden) layers which are considered to be *deep*.

Fully Connected Layers

Fully connected layers are arguably the most simple or basic type of network layer. Neurons are usually stacked in layers (cf. Fig. 2.4 for the multilayer perceptron) where each neuron of a particular layer is connected with each neuron of the preceding layer as shown in Fig. 2.7. In the popular task of image classification, such layers are usually placed behind a block of convolutional layers right before a final Softmax activation layer which transforms the output values of the last layers into a class probability distribution.

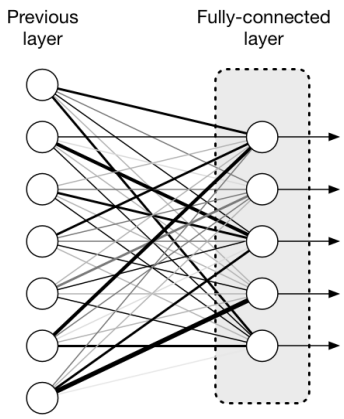


Figure 2.7: A fully connected layer. The thickness of connecting lines represent the magnitude of weights associated with a particular connection.
Source: https://www.medium.com/

Convolutional Layers

Although introduced over two decades ago by [Lecun et al., 1998], convolutional neural networks have gained traction significantly due to successes in the ImageNet competition by [Krizhevsky et al., 2012]. Today, image-based deep learning cannot be imagined without convolutional layers. The core mechanism of those layers is the convolution (pixel-wise summation) of one or multiple kernels (also called filters) with some input image in the first layer or feature map in deeper layers. A convolutional layer's output shape depends on the chosen number of filters and their sizes as well as the amount of padding applied to the input feature map and the stride of the kernel. Refer to Fig. 2.8 for an example with common choices of such parameters. Note that convolutional neural networks are sparse in that every value in an output feature map only depends on a small number of input values (convolved with some kernel). This preserves topological information which would be lost when transforming an image into a one-dimensional vector and feeding it into a fully-connected network. Convolutional layers are also very efficient due to the re-use of parameters since the values of some kernel are constant while sweeping over the input feature map and get updated only after a full training step is done. The input feature map can hold image-like matrices, e.g. a colored image is composed of three channels. A single filter would then produce three output maps which are being added for a final output map. Each filter yields such an output map, thus the number of filters decides for the number of channels in the final output feature map.

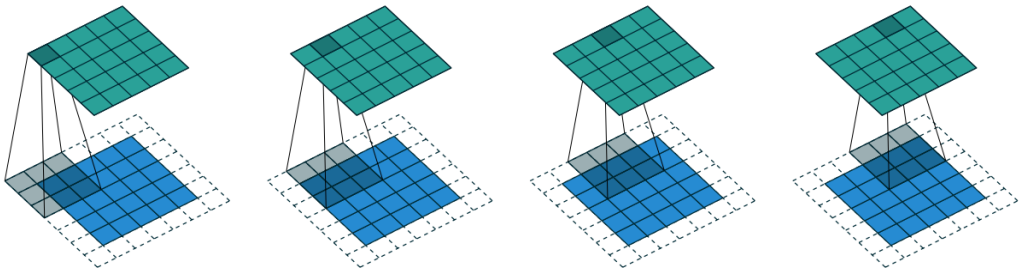


Figure 2.8: A convolutional kernel of size 3×3 (transparent grey) moves along spatial positions on a feature map (blue) with zero-padding of one pixel in both dimensions to produce the output feature map (green) which, in this case, has the same dimensions as the input map.
Source: [Dumoulin and Visin, 2018]

Since the networks used throughout this work fall into the family of Auto Encoders, they follow a encoder-decoder structure, that is, the input image is encoded down to a smaller latent space representation via convolutional layers before being upsampled again with so-called deconvolutional layers, also known as transpose convolutions, whose mechanics are depicted in Fig. 2.9.

Overfitting and Dropout

One common concern with training deep neural networks is that of overfitting to the training data. Models which overfit to the training data do not generalise well to held-out test data, that is, they perform poorly on unseen samples although showing better performance on in-distribution training data. For that reason, one does usually sub-sample a held-out validation set from the training data at hand, which is not used in training, to validate the models performance as training goes on. When the model shows significantly higher performance on the training set compared to the validation set, this is usually a sign that the model does indeed overfit to the training data.

Several strategies exist to tackle this problem: Increasing the amount of training data usually helps to avoid overfitting due to increased diversity in the training data. This process might be

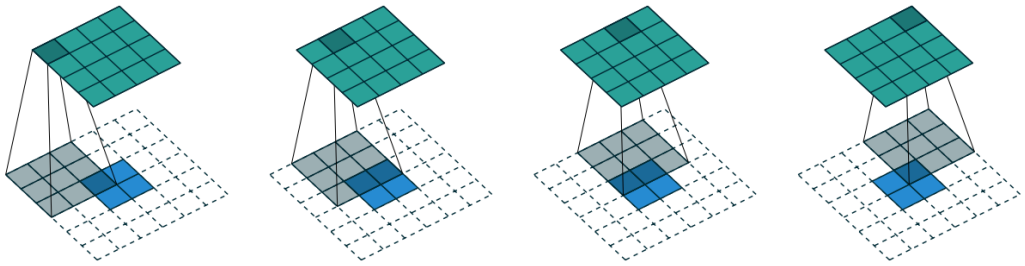


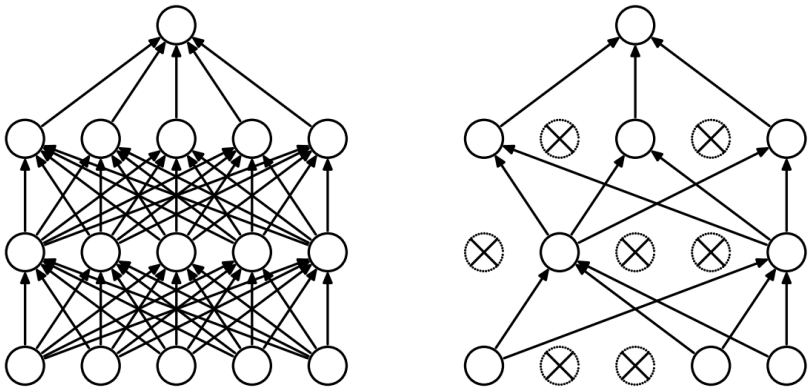
Figure 2.9: A transpose convolutional kernel of size 3×3 (transparent grey) moves along spatial positions on a 2×2 feature map (blue) with zero-padding of two pixels in both dimensions to produce the output feature map (green) whose extent is larger than the input feature map.
Source: [Dumoulin and Visin, 2018]

Figure 2.10: Schematic training and validation loss over the course of ongoing model training. The model starts overfitting to the training data roughly at the epoch corresponding to the dashed vertical line when the loss on the held-out validation data starts to increase. This would be the optimal point to do early stopping of the training to achieve best performance and maximum generalizability.

expensive (e.g. highly paid experts manually labelling images) or infeasible due to the lack of abundance of relevant events. In the setting of unsupervised learning it's not even an option by nature. Those difficulties can be overcome partially by artificially creating new data samples from existing ones by applying data augmentation. That is, applying geometric or lighting transformations to input images, such as random flips, sheers, scalings, crops or exposure adaptations. Data augmentations have been used heavily in image-based deep learning to make models more robust against distributional shifts or deviations from unseen examples compared to the training data distribution.

Dropout [Srivastava et al., 2014] is yet another technique to tackle the issue of overfitting and can be seen as a way to regularize the neural network. Regularization can be seen as a way to prevent unwanted model complexity. It works by dropping neurons with a certain probability at train time, that is, setting them to zero. The dropout rate enters as yet another hyper-parameter which needs to be chosen by the programmer and is usually set to 40-70%. At test time, all neurons are used as is, that is, no dropout is being applied. This procedure is depicted in Fig. 2.11. An intuition as to why this technique would work in the first place can be given as follows. From the

perspective of a single neuron, it might be dangerous to rely on single input features since, with dropout, those input features might be gone in the next iteration. Thus, it is rewarding to spread out the weights which corresponds to shrinking them. This draws an interesting connection to the well-known L2-regularization technique, which is frequently applied to classical machine learning models and aims to shrink a models parameters to reduce its complexity. Dropout has been used extensively with great success in the field of deep learning and particularly in the Computer Vision community since training data is usually sparse compared to the high dimensionality of input samples.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.11: Schematic representation of a fully connected network with two hidden layers and a single output neuron without (a) and with (b) dropout applied.
Source: [Srivastava et al., 2014]

More recently, it could be shown that Dropout can be used to represent the models uncertainty [Gal and Ghahramani, 2016] in its predictions. This simple technique leaves Dropout turned on also during test time and performs several forward passes of a training sample through the network, each time with different random Dropout. The uncertainty of the prediction can then be reported via the standard deviation of the individual predictions.

Batch Normalization

Batch normalization makes neural networks more robust to various sets of hyper-parameters and can also speed up training deep models and has been introduced by [Ioffe and Szegedy, 2015]. Batch normalization works by normalizing the output values (usually after the activation function) of a single layer such that their mean and variance are parameterized by two learnable parameters. Intuitively speaking, batch normalization applies similar normalization to deeper layers, and thus facilitates training, as does normalization (scaling) of input features, a well-known technique used when training machine learning models. In more detail, even though activation values in some hidden layers might change when inputting different samples, their mean μ and standard deviation σ stay the same. Note that μ and σ for a particular layer do not have to be 0 and 1 necessarily. This makes the activation values throughout the network more stable and lets the later layers work with normalized input data much like the very first hidden layer when normalizing the original input data.

It should be noted that batch normalization adds additive (by subtracting the mean) and multiplicative (by dividing through the standard deviation) noise to the layers activations. The noise enters since the normalization is usually performed by only taking the samples from one single mini-batch into account. This in turn has a regularization effect since individual units tend to

rely less on values of single input connections. This raises the question whether one should use Dropout next to batch normalization or rely on the regularization effect of batch normalization itself. Note that for larger batch sizes, regularization due to batch normalization decreases.

2.5 Graphical Models

Graphical Models are probabilistic models which make use of a graph-based representation to express conditional dependencies of a collection of random variables. One particular class of graphical models based on Directed Acyclic Graph (DAG) are known as Bayesian Networks or Belief Networks. At this point, a very simple form of a Belief Network shown in Fig. 2.12 shall be discussed since it is essential to understand the inner workings for VAEs which will be introduced in Sec. 3.1.2. Note that this section relies on shorthand notation, that is, the probability distribution $p(X = x)$ for a random variable X is represented in shorthand notation by $p(x)$.

In this case, we are interested in inferring the probability distribution of the latent variable $p(z)$, which governs the generative process for the data distribution $p(x)$. That is, we want to infer the conditional distribution $p(z|x)$. This can be expressed using **Baye's theorem**

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x,z)}{p(x)} \tag{2.13}$$

In this setting, $p(z|x)$ is known to be the *posterior*, $p(z|x)$ is the *likelihood*, $p(z)$ the *prior* and $p(x)$ the *marginal* distribution. The marginal distribution often renders itself intractable, especially when deploying deep neural networks as function approximators since the integral

$$p(x) = \int p(x|z)p(z)dz \tag{2.14}$$

becomes very high dimensional and intractable. Thus, to perform posterior inference, one employs different techniques like Markov Chain Monte Carlo sampling (MCMC) or Variational Inference. The latter will be introduced in Sec. 3.1.2 as it poses the foundation for VAEs which is the major framework used throughout this work.

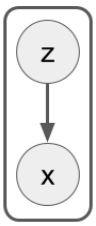


Figure 2.12: An example of a simple graphical model. The *observed* random variable x depends conditionally on the hidden or *latent* variable z which governes the generative process for samples following x.

2.6 Information Theory

Information theory is based on the work and findings of the mathematician Claude Shannon. It is concerned with problems like the handling of information, its compression and transmission. Furthermore Shannon's definition of entropy, which reflects a notion of uncertainty, has become

increasingly important in the field of deep learning when analysing the reliability of model predictions. This chapter shall give a short overview on the most relevant pieces out of this broad field which are relevant to understand techniques and methods used and applied in this work.

2.6.1 Information Content

Let X denote a discrete random variable with a probability density function $p(X = x) = p(x)$. The associated possible outcomes are defined by the alphabet $\mathcal{X} = \{x_1, x_2, ..., x_n\}$. The information content is then defined as

$$I(x) = -\log_2 p(x) \tag{2.15}$$

and describes the surprise of a particular event. Intuitively speaking, due to the nature of the logarithm, very likely events, that is, events with high probability of occurrence, yield low information content and vice-versa.

2.6.2 Shannon Entropy

The Shannon Entropy is defined as the expectation value of the information content

$$H(X) = \sum_{x \in \mathcal{X}} p(x)I(x) = -\sum_{x \in \mathcal{X}} p(x)log_2 p(x) \tag{2.16}$$

and represents the average uncertainty within a random variable. For example, the Bernoulli experiment of tossing a coin has the highest possible value of entropy if the coin is fair, that is $p(X = heads) = p(X = tails) = 1/2$ since $H(X) = -(1/2log1/2 + 1/2log1/2) = 1$, as shown in Fig. 2.13.

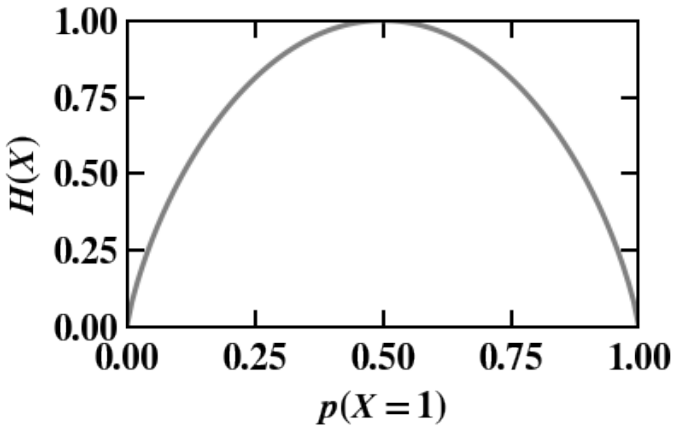


Figure 2.13: The Shannon entropy as a function of probability p that some positive event will occur for the case of a Bernoulli experiment. The negative probability is then given by $p - 1$. The highest entropy and likewise highest uncertainty with respect to the outcome is given when $p = 1/2$.

2.6.3 KL Divergence

The KL divergence represents a measure of dissimilarity for two probability distributions. It tells us how strong two distributions $p(x)$ and $q(x)$ differ from one another. It is defined as

$$KL(p||q) = \sum_{x \in \mathcal{X}} p(x) log \frac{p(x)}{q(x)} = -\sum_{x \in \mathcal{X}} p(x) log \frac{q(x)}{p(x)} \tag{2.17}$$

for the discrete and

$$KL(p||q) = \int_{-\infty}^{+\infty} p(x) log \frac{p(x)}{q(x)} dx \tag{2.18}$$

for the continuous case. In the discrete case (Equ. 2.17), the latter representation is often preferred in the literature. The KL divergence is non-negative and not symmetric:

1. $KL(p||q) \geq 0$
2. $KL(p||q) \neq KL(q||p)$

The KL divergence will come into play in Sec. 3.1.2 where it shows up as a term in the loss function of the Variational Auto Encoder framework. Fig. 2.14 shows an example of the behaviour of the KL divergence in the case of two simple discrete probability distributions to gain some intuition.

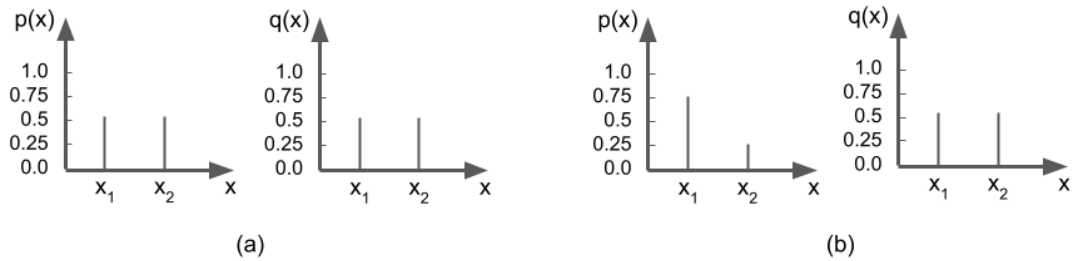


Figure 2.14: Two examples of discrete distributions and $p(x), q(x)$ and the respective KL divergence between them. (a) $KL(p||q) = 0$ due to identical distributions and (b) $KL(p||q) = {}^{3}/_{4} log({}^{3}/_{2}) + {}^{1}/_{4} log({}^{1}/_{2}) > 0$ indicates dissimilar distributions

2.7 Metrics

2.7.1 Evaluation of a Binary Classifier

Since binary classification is of great concern for the experiments and results presented throughout this work, this chapter shall present the most important concepts and evaluation strategies when using binary classifiers. Binary classification is concerned with predicting an input sample to belong to a positive (usually denoted with label 1) or negative class (label 0). This results in four types of predictions. True positives (TP), that is, a sample belongs to the positive class and the model assigns a positive label to the sample. True negatives (TN), that is, a sample belongs to the negative class and the model assigns a negative label to the sample. False positives (also known as type 1 error, FP), that is, the sample actually belongs to the negative class, but the model predicts that it belongs to the positive class. And finally, false negatives (type 2 error, FN), that is, the sample actually belongs to the positive class but the model predicts it to belong to the negative class. The term positive and negative has no valuation and it is up to the programmer to decide which class represents the positive and negative samples depending on the task at hand. In this work, the positive class is usually assumed to be the *lesional* or *OOD* class.

The Confusion Matrix

Fig. 2.15 arranges the aforementioned categories into a 2×2 confusion matrix, a common tool to evaluate the performance of a binary classifier.

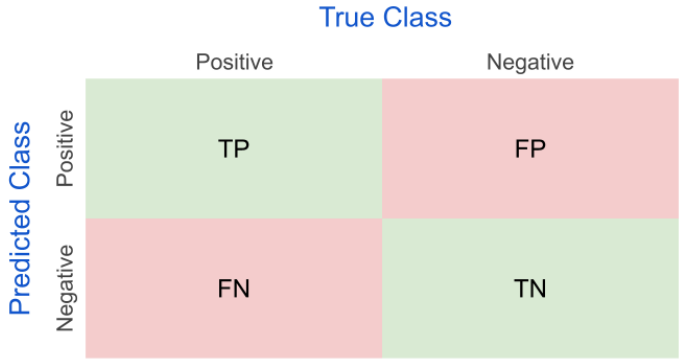


Figure 2.15: Confusion matrix for the binary classification problem.

Accuracy

Accuracy is frequently used as a summarizing metric for a binary classification problem. It gives the ratio of all correct predictions over all predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.19}$$

It should be used with great care since it can give very skewed impressions of the performance of a classifier when dealing with unbalanced datasets. For example, a classifier which predicts all samples to be diseased (positive class) given a set of $10'000$ blood samples while there are 100 sick patients achieves an accuracy of 99% which is not expressing the fact that it is completely failing in detecting the diseased samples while at the same time marking healthy samples as such.

Precision and Recall

Precision answers the question "From all the samples that have been predicted to be positive, how many are actually positive?".

$$precision = \frac{TP}{TP + FP} \tag{2.20}$$

Recall on the other hand answers the question "What fraction of positive examples has the classifier been able to detect as such?".

$$recall = \frac{TP}{TP + FN} \tag{2.21}$$

Reporting precision and recall compared to only accuracy is often a much stronger statement compared to stating only the accuracy of a classifier. For example, the classifier in the example above with an accuracy of 99% would achieve a recall of 100% (since all positive, diseased samples, have been detected) but a precision of only 1%. This reveals the fact that although the classifier is being able to detect all diseased samples and thus has the highest number of true positives possible, it produces a massive amount of false positives such that we cannot really trust its predictions. In practice, there is a trade-off between precision and recall which is expressed in the Precision-Recall Curve (PRC), discussed in the next section.

ROC and PRC Curves

A classifiers prediction usually comes in the form of a probability distribution or with a more generic scoring function value. For example, for an input blood sample, the classifier would report with 20% probability that the sample is diseased and consequently with 80% probability that it is not diseased. It is then the operators choice to set a threshold on the predictive probability at which a sample should be considered diseased or not. By adjusting the threshold to low probabilities, the recall increases since a new sample is marked positive already with a low predicted probability or scoring. On the other hand, the precision decreases since more false positive are being introduced. There exist two common ways to visualize and summarize those findings in a single metric.

The **ROC Curve** (Receiver Operating Curve) plots the True Positive Rate (TPR) against the True Negative Rate (TNR). The True Positive Rate is another denotation for the recall and expresses the very same thing. The True Negative Rate or specificity is defined by

$$TNR = \frac{TN}{TN + FP} \tag{2.22}$$

and answers the question "What is the fraction of actual negatives samples, which have been labeled as such by the classifier?".

The **AUC** (Area under the ROC curve) summarizes the overall performance of a binary classifier by simply reporting the area under the curve for a given classifier. A high value of AUC means that the classifier performs well, that is, it generates high recall while keeping the false positive rate low, over a large range of thresholds. A random classifier (random guessing) yields an AUC of 0.5 while a perfect classifier yields an AUC of 1.0. A classifier which systematically predicts the wrong class will yield an AUC smaller than 0.5. Fig. 2.16 shows a ROC curve for three different classifiers. The blue classifier is closest to the hypothetical perfect classifier in the upper left and has the overall best performance across all probability thresholds with the largest AUC. Note that the operater can choose an operating point (probability threshold) depending on the goal of the predictive system. If it is very important to capture close to all of the positive examples, he will set the threshold rather low to achieve a high recall with the drawback of introducing more false positives. In that case, he will move on the ROC curve to the upper right. If however it is important to be very sure that a positive prediction is actually positive, he would chose a high threshold and move to the bottom left to reduce the false positive rate.

The **PRC Curve** (Precision-Recall Curve) depicts the trade-off between precision and recall introduced earlier. It is an alternative to the ROC Curve and frequently used to visualize the overall performance of a binary classifier.

Similar to the AUC, the **AUPRC** (Area under the Precision-Recall Curve) summarizes the performance of a classifier in a single metric.

Scientific papers in the field of Unsupervised Anomaly Detection and OOD detection typically report AUC and/or AUPRC scores to describe the performance of the classification models. This could concern the ability of an anomaly detection model to detect lesional MRI brain scans or the ability of an OOD detection framework to detect distributional shifts of input data samples. Fig. 2.16 give a schematic overview of a ROC and PRC curve with corresponding annotations of the principles discussed above.

2.7.2 Image Segmentation

Unsupervised Lesion Detection is concerned with the task to segment out pixels from an input image which belong to lesional structures such as brain tumors. This is similar to the the binary classification problem on a pixel level. The community of image segmentation has developed their own metrics for this type of tasks. The most common metrics reported are the Intersection over Union (IoU) and Dice score introduced up next.

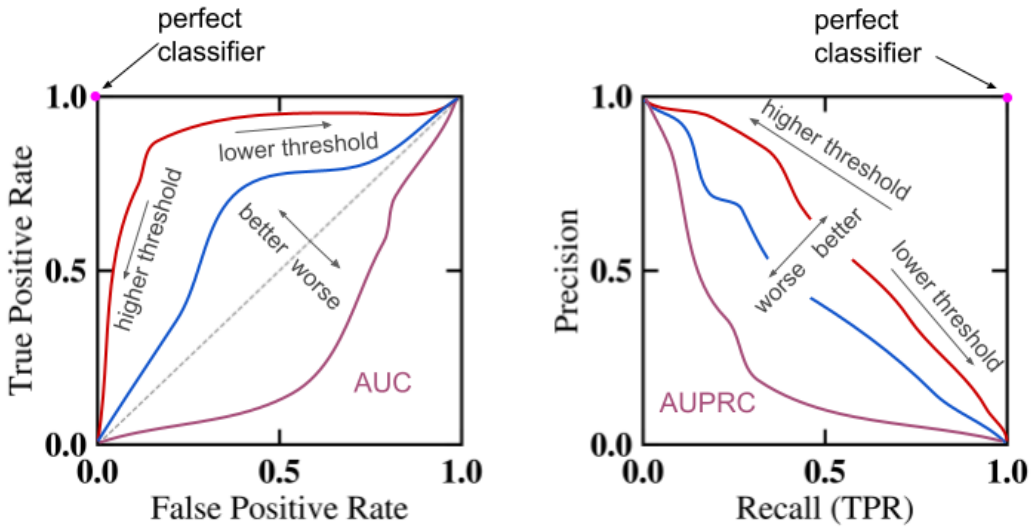


Figure 2.16: **Left:** ROC curve comparing three classifiers. The hypothetical perfect classifier is in the upper left with 100% TPR and 0% FPR. The classifier belonging to the red curve has the overall highest AUC (area under ROC curve) and is therefore preferable to the other two. The purple classifier has an AUC smaller than 0.5 which indicates systematically wrong predictions. The horizontal dashed center line represents a random classifier with an AUC of 0.5. **Right:** PRC curve comparing three classifiers. The hypothetical perfect classifier lies on the upper right with 100% recall, meaning all positive instances have been detected and 100% precision, meaning that all instances predicted to be positive are actual positives. The direction of movement along either the ROC or PRC curve when changing the scoring threshold is indicated with arrows.

Intersection over Union (IoU)

The IoU metric is defined as

$$IoU = \frac{|target \cap prediction|}{|target \cup prediction|} \tag{2.23}$$

where $target$ and $prediction$ denote the set of pixels belonging to the ground truth and prediction mask belonging to the class of pixels one wants to segment. The IoU values lie in the range $[0, 1]$ where a value of 0 means that the intersection of prediction and ground truth is zero, that is, no pixel could be segmented successfully, and a value of 1 denotes perfect overlap of the two and represents the highest score possible.

Dice Score

The Dice score is similar to the IoU metric and reads

$$Dice = \frac{2 * |target \cap prediction|}{|target| + |prediction|} \tag{2.24}$$

Just like the IoU metric, the Dice score ranges from 0 (worst) to 1 (best). Note that instead of the union, the denominator now holds the sum of the target and prediction areas. The factor 2 in the nominator thus makes sure that the Dice score sums up to 1 for a perfect prediction, similar to the IoU. While there are subtle differences in the behaviour of those metrics when it comes to taking averages over many inferences, that is, IoU measures closer to the mean performance while Dice measures closer to the worst case performance, their application depends mainly on what is commonly used in the respective domain. In the field of Unsupervised Lesion Detection on brain MRI, the predominant metric reported is the Dice score.

2.8 Magnetic Resonance Imaging (MRI)

The methods presented throughout this work are first and foremost applied to MRI data which is why this imaging modality shall be introduced in this section in more detail. And possibly also due to the fact that this thesis concludes my studies of Physics which I would like to highlight briefly in this chapter.

Physics Background Nuclei contained in the human body might carry spin **S** which translates into magnetic moment $\boldsymbol{\mu}$ via the gyromagnetic ratio γ

$$\boldsymbol{\mu} = \gamma \cdot \mathbf{S} \tag{2.25}$$

While the proportionality factor γ depends on the specific nuclei in consideration, it should be noted that a high value of γ is usually preferred in the context of MRI since this would translate a given "amount" of spin into a large amount of magnetic moment. This is key in obtaining strong MRI signals for detection. In a world of very small length scales, one has to consider quantum mechanical effects which reveal that spins (and energy levels) occur in quanta. The z-component of a nuclei spin vector **S** is defined by

$$S_z = \hbar m \tag{2.26}$$

where $\hbar = 1.054 \times 10^{-34} J \cdot s$ is the reduced Planck's constant and m the magnetic quantum number

$$m = -I, -I+1, ..., I-1, I \tag{2.27}$$

while I is the spin quantum number. It reads $I = 1/2$ in the case of ${}^1H$, thus $m = \pm 1/2$ which results in $\mu_z = \gamma\hbar m = \pm 1/2 \hbar\gamma$. For simplicity, the following considerations will always consider the case of ${}^1H$ isotopes. Hereby, the z-component is defined through the direction of an externally applied magnetic field **B**. Strong alignment of the magnetic moment $\boldsymbol{\mu}$ with **B** results in a low-energetic state and vice-versa. Note that full alignment can never be achieved due to quantum mechanical constraints, since

$$|\mathbf{S}| = \hbar\sqrt{I(I+1)} \tag{2.28}$$

The energy content in a given spin-state is defined by

$$E_m = -\boldsymbol{\mu} \cdot \mathbf{B} = -\mu_z \cdot B_0 \tag{2.29}$$

where the last equality takes into account that the external magnetic field **B** is perfectly aligned in the z-direction by definition. This reveals that the $m_{-1/2}$-state is high-energetic ($E_m = {}^{+\hbar}/_2 B_0$) while $m_{+1/2}$-state is lower energetic ($E_m = {}^{-\hbar}/_2 B_0$).

MRI relies on the fact that for a given isotope, that is, an element defined by it's number of protons with possibly varying number of neutrons, in the human body, e.g. ${}^1H$, the number of nuclei residing in a low-energetic up-spin state, $n_{-1/2}$ and those in a high-energetic low-spin state, $n_{+1/2}$, differs and follows the so-called Boltzmann statistics

$$\frac{n_{-1/2}}{n_{+1/2}} = exp(-\frac{\Delta E}{k_B \cdot T}) \tag{2.30}$$

where $k_B = 1.38064852 \times 10^{-23} m^2 kg s^{-2} K^{-1}$ is the so-called Boltzmann constant, T is the temperature in Kelvin and ΔE represents the energy-gap between high and low energetic nuclei. Note that this gap depends linearly on the background field strength B_0, a key factor exploited by MRI systems since this field strength can be varied over time and space with the appropriate facilities.

Nuclei shifting from a high to a lower energetic state release photons with energy

$$\Delta E = \hbar \cdot \omega_L \tag{2.31}$$

where ω_L is known to be the larmor frequency. MRI systems measures the decay of macroscopic magnetization which is defined by

$$M_0 = \sum \mu_z \tag{2.32}$$

for a given volume of interest over which we consider the sum of magnetic moments. This macroscopic view allows us to consider a classical approach and observe the magnetization dynamics which are governed by the so-called Bloch equations

$$\frac{d}{dt}\mathbf{M} = \gamma \mathbf{B} \times \mathbf{M} - \begin{pmatrix} {}^{M_x}/_{T_2} \\ {}^{M_y}/_{T_2} \\ (M_z - M_0)T_1 \end{pmatrix} \tag{2.33}$$

In matrix-vector form they read

$$\frac{d}{dt}\mathbf{M} = \begin{pmatrix} {}^{-1}/_{T_2} & -\gamma B_z & \gamma B_y \\ B_z & {}^{-1}/_{T_2} & -\gamma B_x \\ -\gamma B_y & \gamma B_x & {}^{-1}/_{T_1} \end{pmatrix} \mathbf{M} + \begin{pmatrix} 0 \\ 0 \\ {}^{M_0}/_{T_1} \end{pmatrix} \tag{2.34}$$

MRI principles Now, consider adding a circularly polarized field with circular frequency w_{RF}, which yields non-zero B_x and B_y components of the resulting field which will now precess around to z-direction. To complicate things further, the Bloch equations reveal that the total magnetization vector $\mathbf{M}$ itself precesses around the net magnetic background field. Thus adding such a secondary field with an appropriate frequency perturbs the magnetization and changes the transverse and longitudinal magnetization components. After shutting down the radio-frequency pulse those magnetization components get relaxed, meaning they decay exponentially. The time horizon for transverse relaxation is what we denote T_2 relaxation while T_1 is associated with longitudinal relaxation. Those are typically in the range of $40 - 100$ msec for T_2 and $200 - 900$ msec for T_1. The key observation to obtain tissue contrast in MRI images is the fact that the relaxation times differ for different types of tissue.

Magnetic moments induce magnetic fields and signal detection can be carried out by placing a detector coil close to the patient's body and exploiting the Farraday's induction principle to measure an induced voltage in the coil.

Position localization of the signal is done using so-called gradient fields which are put in place with so-called Golay pairs. They introduce magnetic field gradients ${}^{dB_z}/_{dz}$, ${}^{dB_y}/_{dz}$, ${}^{dB_x}/_{dz}$ which can be used for position localization using

$$\omega(z) = \gamma(B_0 + \frac{dB_z}{dz}z) \tag{2.35}$$

which reflects that fact that gradient fields make the emitted photon frequency during relaxation position dependent.

T1- T2-weighted imaging While early MRI images basically exploited the mechanisms explained above, also known as frequency encoding, and produced images reflecting the spin distribution within the body, more modern approach are often weighted by the relaxation times T_1 and T_2 of tissues. While there are different methods to obtain T_1 weighted images, one variant is to apply a $90\,\mathrm{deg}$ flip of magnetization and let one tissue type relax by waiting for the so-called recovery time. After a second RF pulse, there will be more transverse magnetization available

in this tissue for imaging which will highlight tissue with shorter T_1 relaxation times when the recovery time is chosen to be small. T2-weighted images are somewhat more complicated and obtained by so-called Spin-Echo experiments where a $90\,\mathrm{deg}$ pulse is followed by a $180\,\mathrm{deg}$ pulse before imaging. Fig. 2.17 shows axial slices from T2- and T1-weighted MRI brain scans.

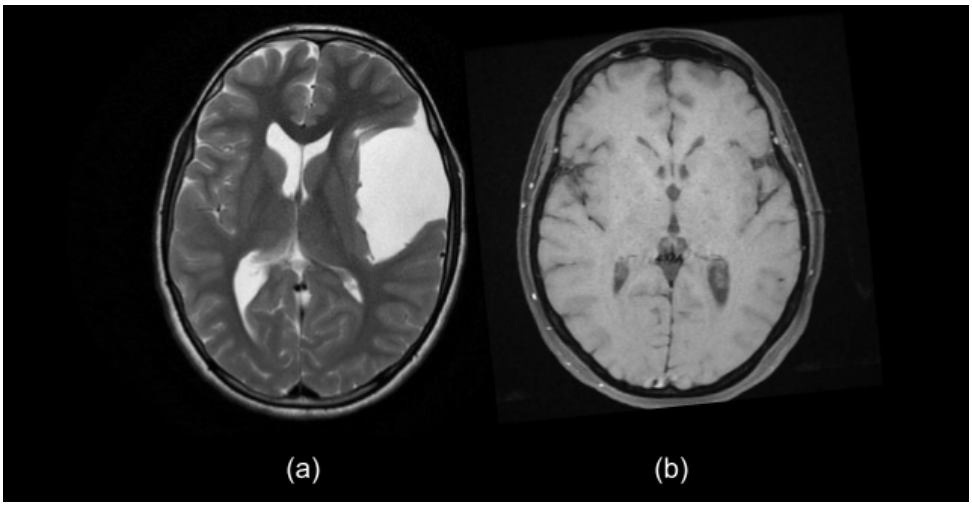


Figure 2.17: (a) A T2-weighted axial-slice MRI brain scan with a hyperintense lesion on the upper right. (b) A healthy T1-weighted slice.
Source: Case courtesy of Assoc Prof Frank Gaillard, Radiopaedia.org, rID: 28272

Chapter 3

Related Work

3.1 Generative Models

The basic concepts behind generative modelling have been introduced in Sec. 2.2.2 while graphical models have been discussed in Sec. 2.5. In the following, a number of relevant Deep Learning based generative models shall be introduced to understand how they can be used for the task of unsupervised anomaly detection. In the present case of segmenting anomalies within brain MR images, we use the notion of deep unsupervised lesion detection. The generative models outlined in this section provide the foundation for this intend.

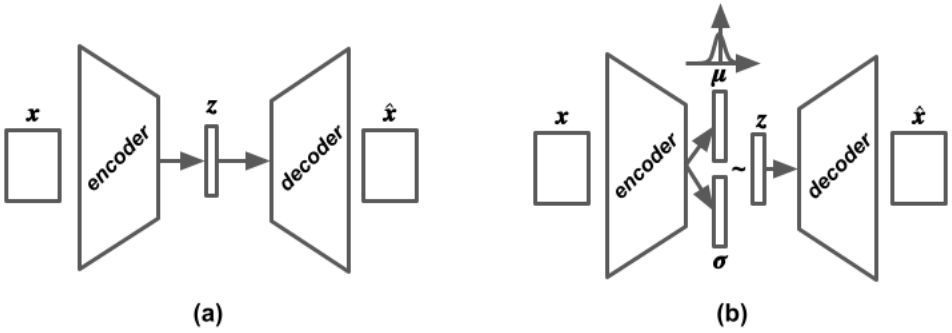


Figure 3.1: Schematic depictions of (a) Auto Encoder (AE) and (b) Variational Auto Encoder (VAE) architectures. Both frameworks *encode* an input sample $\mathbf{x}$ into a latent space representation $\mathbf{z}$ before *decoding* it back into sample space to obtain a reconstruction $\mathbf{x}$. While the vanilla AE architecture is completely deterministic, the VAE framework models the latent space representation by means of a probability distribution parametrized by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. An actual latent sample $\mathbf{z}$ is then being sampled from this distribution from which the decoder will eventually generate a reconstructed sample.

3.1.1 Autoencoder (AE)

An Autoencoder learns to map a given input $\mathbf{x}$ using an encoder network $enc_\theta(\mathbf{x})$ with parameters θ into a so-called latent space representation $\mathbf{z} = enc_\theta(\mathbf{x})$ before reconstructing the original input via a decoder network dec_ϕ as $\hat{\mathbf{x}} = dec_\phi(enc_\theta(\mathbf{x}))$. It does so by minimizing the reconstruction error $|\mathbf{x} - \hat{\mathbf{x}}|_p$, where p is usually chosen to be 1 (absolute error) or 2 (squared error) in the context of image reconstruction. The latent space representation can be either a one-dimensional vector or a higher-dimensional, tensor-like representation. However, in practice it is usually chosen to be low-dimensional compared to the input data to force the learning of a meaningful representation

of the most important features of the input data due to limited amount of space available. This can be interpreted as dimensionality reduction while the most important features of the data are preserved to be able to create a meaningful reconstruction. In fact, in can be shown that a simple AE with one hidden layer is equivalent to PCA introduced in Sec. 2.1, arguably the most popular dimensionality reduction technique. A schematic depiction of an image-based Autoencoder with convolutional hidden layers and a one-dimensional bottleneck is shown in Fig. 3.1 (a). Vanilla Autoencoders do not impose any constraints or regularization on the latent space representation, allowing any configurations and heavy imbalances in terms of occupied volume for different types of samples, which might not be beneficial. The Variational Autoencoder framework in upcoming section addresses this issue by imposing a regularization term on the latent space representation.

3.1.2 Variational Auto Encoders (VAE)

The family of Variational Auto Encoders are the major type models used throughout this work justifying the somewhat in-depth introduction in the following section. In a nutshell, Variational Auto Encoders [Kingma and Welling, 2014] introduce two distinct changes to the Auto Encoder framework introduced in Sec. 2.2.2 even though their development originates from a very different point of view. First, the loss function is being adapted by introducing a term based on the KL-divergence (cf. Sec. 2.6.3). The additional loss term introduces regularization into the latent-space distribution which means a more expressive latent code. Second, the latent space representation changes from a tensor-like layer structure to a layer representing a probability distribution. Finally, by introducing stochasticity into the framework, the so-called reparameterization trick is being deployed to enable render training such a network feasible.

Consider the graphical model introduced in Fig.2.12. The generative process is given by $p(z)p(x|z)$. The posterior $p(z|x)$ however is intractable as stated before. The idea is then to approximate $p(z|x)$ by using a tractable distribution $q(z|x)$. For simplicity, we denote to this distribution as $q(z)$ here but the general case still holds. For it to be a decent approximation, we want it to be close to $p(z|x)$ which is achieved by minimizing the KL divergence

$$\begin{aligned} KL(q(z)||p(z|x)) &= -\sum_z q(z) \log \frac{p(z|x)}{q(z)} \\ &= -\sum_z q(z) \log \frac{\frac{p(x,z)}{p(x)}}{q(z)} \\ &= -\sum_z q(z) \log \frac{p(x,z)}{q(z)p(x)} \\ &= -\sum_z q(z) \left[log \frac{p(x,z)}{q(z)} + \log \frac{1}{p(x)} \right] \\ &= -\sum_z q(z) \log \frac{p(x,z)}{q(z)} \sum_z q(z)(x) \\ &= \log p(x) - \sum_z q(z) \log \frac{p(z,x)}{q(z)} \end{aligned} \tag{3.1}$$

Rearranging the result above yields

$$\log p(x) = KL(q(z)||p(z|x)) + \underbrace{\sum_z q(z) \log \frac{p(x,z)}{q(z)}}_{\mathcal{L}} \tag{3.2}$$

Note that $\log p(x)$ is a constant when varying $q(z)$, therefore minimizing the KL divergence $KL(q(z)||p(z|x))$ corresponds to maximizing $\mathcal{L}$, also known as the *Evidence Lower Bound (ELBO)* since it acts as a lower bound for $\log p(x)$. $\mathcal{L}$ is a true lower bound of $\log p(x)$ due to the fact that the KL term is strictly non-negative, thus $\mathcal{L} < \log p(x)$. Since the logarithm is a strictly monotonically increasing function, maximizing $\mathcal{L}$ also has the effect of generating the observation x since we increase $p(x)$.

Some further transformations yield the more common form of the ELBO

$$\begin{aligned}
\mathcal{L} &= \sum_z q(z) \log \frac{p(z,x)}{q(z)} \\
&= \sum_z q(z) \log \frac{p(x|z)p(z)}{q(z)} \\
&= \sum_z q(z) \left[\log p(x|z) + \log \frac{p(z)}{q(z)} \right] \\
&= \underbrace{\sum_z q(z) \log p(x|z)}_{\mathbb{E}_{q(z)} \log p(x|z)} + \underbrace{\sum_z q(z) \log \frac{p(z)}{q(z)}}_{-KL(q(z)||p(z))} \\
&= \mathbb{E}_{q(z)} \log p(x|z) - KL(q(z)||p(z))
\end{aligned} \tag{3.3}$$

The final objective of the vanilla VAE framework is to maximize this lower bound. This is commonly achieved by minimizing the negative value if $\mathcal{L}$ which is comprised of two terms, the *reconstruction error* and the *KL divergence between the approximate posterior and the prior* which we denote as the *prior loss* in this context.

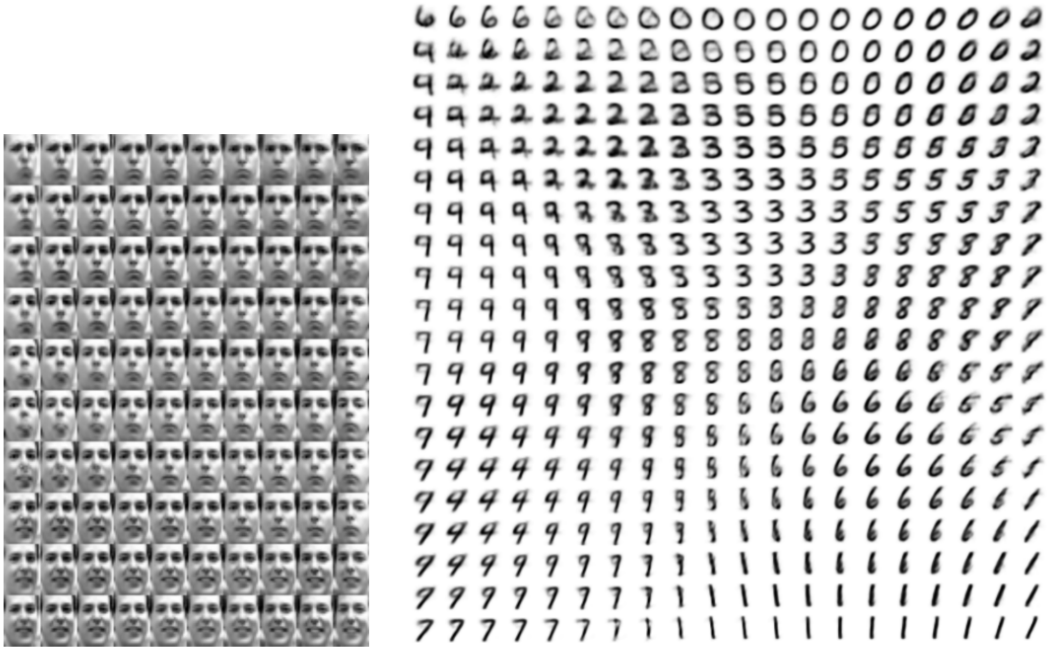
$$\log p(\mathbf{x}) \geq \mathcal{L} = \underbrace{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Error}} - \underbrace{D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]}_{\text{Prior Loss}}, \tag{3.4}$$

While the first term aims to reduce the reconstruction error (similar to the standard Auto Encoder framework), the second term minimizes the KL-divergence between some chosen prior distribution $p(z)$ and the approximate posterior distribution $q(z)$ or $q(z|x)$ to be precise. In practice, the first term is often replaced by $|\mathbf{x} - \hat{\mathbf{x}}|_p$ with $p \in \{1, 2\}$. There is no way to make a well-informed decision on what distribution to chose for the latent space prior distribution $p(z)$. Hence, in practice, usually a multi-variate Gaussian is chosen for that purpose. In a neural network setting, the transformations $q(z|x)$ and $p(x|z)$ are represented by an encoder enc_θ with parameters θ and a decoder dec_ϕ network with parameters ϕ respectively. Fig. 3.2 shows that VAEs are able to learn a meaningful latent space representation which might encode high-level semantic properties of the training data.

The Reparameterization Trick

Without loss of generality, consider the case where an encoder enc_ϕ maps an input image $\mathbf{x}$ to an approximate latent space distribution $q(\mathbf{z}|\mathbf{x})$. When passing a sample through the network, an actual latent representation is being sampled from this latent distribution. The stochasticity introduced by the sampling operation renders the network not differentiable with respect to the loss anymore which is a key requirement for backpropagation to work. Thus, [Kingma and Welling, 2014] proposed the use of the so-called reparameterization trick which, for this particular case, states that the reparameterized latent space representation becomes

$$\mathbf{z} = \boldsymbol{\mu} + \epsilon \cdot \boldsymbol{\sigma} \tag{3.5}$$



(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 3.2: Images are being created by traversing the latent space while varying two distinct dimensions. Samples are being drawn at those various positions and reconstructed by the decoder. (a) The VAE was able to learn the direction of gaze as well as the mood of a person which it encoded into the latent space representation. Trained on the Frey Face Dataset. (b) Similar principle applied to the MNIST [LeCun and Cortes, 2010] dataset which reveals the continuous nature of the latent space encoding.
Source: [Kingma and Welling, 2014]

where $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ which let's us approximate the normally distributed latent space distribution with another normal distribution. Note that the network is now differentiable since $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are deterministic outputs of the encoder network enc_ϕ. The python pseudo-code for a simple forward pass within a VAE is thus given by

```
mu, sigma = encoder(input_image)
latent_code = reparameterize(mu, sigma)
reconstruction = decoder(latent_code)
```

3.2 OOD Detection Using Generative Models

OOD detection is concerned with the task of detecting input samples during inference on a trained model which do not fit the so-called in-distribution data that has been used during training. More specifically, consider the simple case of a discriminative model which infers a predictive distribution $p(y|\mathbf{x})$ for some input $\mathbf{x}$ which at train time is sampled from a training distribution $p(\mathbf{x})$. At test time, inference is run on samples from following a distribution $q(\mathbf{x})$. The objective of OOD detection is then to answer the question whether $p(\mathbf{x}) \equiv q(\mathbf{x})$, that is whether the test set distribution matches the training distribution.

As introduced in Sec. 2.2.2, generative models model the density either implicitly or explicitly. More specifically, in the case of VAE's, the *ELBO* $\mathcal{L}$ acts as a a lower bound for the data

log likelihood $\log p(\mathbf{x})$ and the true likelihood can be obtained via MCMC sampling techniques. Thus it stands to reason to define a single-sided likelihood threshold by running inference on in-distribution training data. All samples which have lower likelihoods during inference compared to this threshold would be regarded as OOD. While this technique has been introduced in the early 90's of the last century [Bishop, 1994], only recently it has been reported that it cannot be trusted when working with high-dimensional input spaces, such as images, which might contain thousands or millions of pixels. Recently, [Nalisnick et al., 2019a] and [Choi et al., 2019] have reported the observation that OOD samples might be more likely for generative models compared to in-distribution training data, which is difficult to make sense of intuitively. This behaviour is shown in Fig. 3.3.

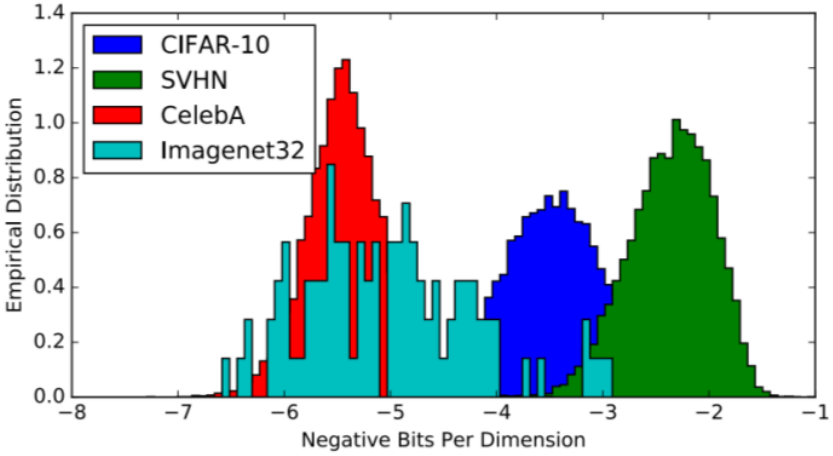


Figure 3.3: This work reveals empirically that a generative model (GLOW in this case [Kingma and Dhariwal, 2018]), might assign higher likelihoods to OOD samples from the Street View House Numbers (SVHN) Dataset [Netzer et al., 2011] when being trained on CIFAR-10 data [Krizhevsky, 2009], which thus is regarded as in-distribution for this experiment. Note that *Negative Bits Per Dimension* can be converted into a likelihood measure.
Source: [Choi et al., 2019]

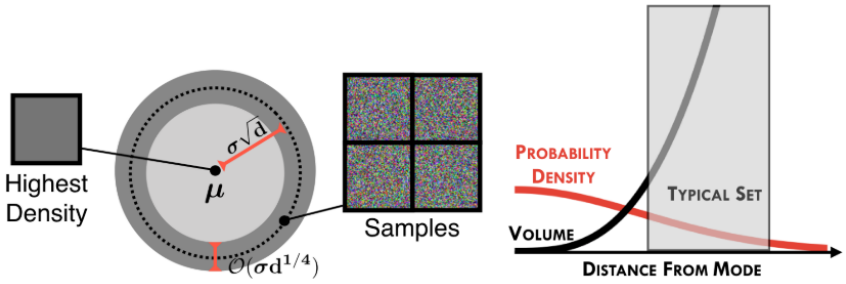
In the same work, [Choi et al., 2019] introduced the so-called WAIC score which consists of the log likelihood and the corresponding variance over an ensemble of model predictions and reads

$$WAIC(\mathbf{x}) = \mathbb{E}_\theta[\log p_\theta(\mathbf{x})] + Var_\theta[\log p_\theta(\mathbf{x})] \tag{3.6}$$

for a model parameterised wit parameters θ. The WAIC score is essentially a corrected version of the expected log likelihood. The correction term subtracts the variance to penalize samples which are sensitive to a particular choice of model parameters. Note that the expectation is taken over multiple models with differing initializations which not only requires large efforts to train these but also slower inference since a single WAIC score prediction needs several inference passes on the various models. While this approach seems to be more robust in various pairings of in-distribution and out-of-distribution datasets, no conclusive explanation as to why that is could be provided. Nevertheless, by combining the notions of density estimation and uncertainty estimation into one metric it does perform well empirically.

Similar findings were reported simultaneously by [Nalisnick et al., 2019a]. As a consequence, a number of works attempted to explain and overcome this counter-intuitive behavior. In a follow-up work [Nalisnick et al., 2019b] reported that there are dataset combinations for which the likelihood distributions of in- and out-of-distribution datasets (training on CelebA, testing on Cifar) actually overlap near perfectly making the situation even more challenging. At the same

time they provided a hypothesis as to why the likelihood method fails using the example of a high-dimensional Gaussian. For a high-dimensional, symmetric Gaussian, the vast majority of the probability mass resides in an annulus centered at a distance $\sqrt{d}$ around the origin (which at the same time represents the mean), where d is the dimensionality of the distribution. Or in other words, when sampling from this high-dimensional Gaussian, the typical samples received will very likely lie in this region and not close to the mean where one would find samples with the highest likelihood given the probability density function. This behaviour is depicted in Fig. 3.4. Following this assumption, they derive a statistical test for typical test set membership and apply this metric to the task of OOD detection.



(a) Gaussian Example (b) Illustration

Figure 3.4: (a) Shows the example of a high-dimensional Gaussian with mean μ and standard deviation σ. This two dimensional represents that while the mean location μ has indeed the highest likelihood for a Gaussian, the *typical set* of this distribution lies in an annulus $\sigma\sqrt{d}$ from which region actual Gaussian noise can be sampled. This annulus region is where most of the probability mass (volume) resides as shown in sub-figure (b).
Source: [Nalisnick et al., 2019b]

[Ren et al., 2019] proposed an OOD detection method based on so-called likelihood ratios. The rational behind this approach is the assumption that any sample might consist of a background component holding generic noise and a foreground component which holds semantic information. By training a *background* model with perturbed training data (assuming perturbations destroy semantic components) along with a foreground model with using original non-perturbed input training data, the are able to correct the likelihood measure by subtracting the background component which they assume to overshadow the semantic components in the standard likelihood metric. By doing so, they are able to achieve a more robust OOD detection framework for various dataset pairings.

Recently, [Morningstar et al., 2020] introduced another density-based OOD detection framework by aggregating various inference statistics, e.g. the reconstruction errors, into the so-called Density of States Estimation (DoSE) score. Specifically, they fit a Kernel Density Estimator (KDE) to each statistics distribution evaluated on the training data and mark novel samples as OOD by thresholding their sum of likelihoods under said estimators. A statistic is to be understood in the sence that a trained model receives input samples on which inference is being performed. Any generated output or intermediate result from the model can be used to deduce a quantity out of which a complete statistic can be collected by running inference on a whole set of samples. There are two major advantages compared to other works described above. 1) the decision boundary is defined through a KDE fit rather than a single-sided threshold on some criteria and 2) it leaves freedom of choice as to which statistics to include in the metric to the implementation. On the other hand this introduces ambiguity as to which statistics might perform well for the task at hand. They have shown however, that there is no significant performance penalty by adding a multitude of training statistics into the framework. The final $DoSE_{KDE}$ score which is then used as a metric

to perform OOD detection reads

$$DoSE_{KDE} = \sum_{j}^{m} KDE_j(\mathbf{x}) \tag{3.7}$$

where KDE_j is a Gaussian kernel density estimation fit on a particular statistic j on the training data. Note that unlike the $WAIC$ score, high $DoSE_{KDE}$ values indicate in-distribution samples.

3.3 Unsupervised Lesion Detection

Lesion detection for MRI brain scan images (or any other type of images for that matter) can be performed on a per-pixel level resulting in the task of semantic segmentation or on a sample-level which is a task of binary classification.

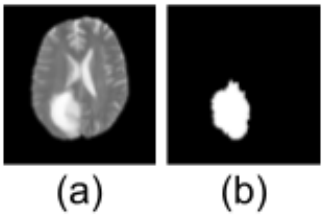


Figure 3.5: (a) Axial slice of an MRI T2-weighted brain scan from the BraTS2017 dataset [Menze et al., 2014] showing a glioma on the bottom left. (b) The corresponding ground truth pixel-level annotation crafted by domain experts.

Prior to the rise of Deep Learning based approaches, unsupervised lesion detection algorithms were proposed by registering images to a healthy standardized brain Atlas and fitting mixture models, amongst others, based on tissue-specific densities to detect lesions [Kamber et al., 1995, Van, 2001, Prastawa et al., 2004] as model outliers.

More recently, Deep Generative models based on Variational Autoencoders (VAE) [Kingma and Welling, 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] have become popular due to their abilities to model high dimensional distributions, essentially learning what is referred to as the normative data distribution. During inference, anomaly detection is then performed by assessing the deviation of test samples from the training distribution. One common way within VAE-based frameworks is to do this via the reconstruction error of the reconstructed to the input sample. For images, pixel-level anomaly detection is performed by thresholding the residual map between an input and reconstruction images, as shown in Fig. 4.1, which builds upon the assumption that lesional regions are expected to have high reconstruction errors since they deviate from the training distribution. Recently, different metrics for pixel-level anomaly detection have been proposed by [Zimmerer et al., 2020]. [Baur et al., 2019] introduced a VAE based framework incorporating adversarial training to improve the realism of reconstructions and avoid memorization. [Chen and Konukoglu, 2018] identified the lack of and improved latent space consistency by adding a regularizing constraint. [Chen et al., 2020] used a VAE with mixtures of Gaussian in the latent space, which is a more expressive prior distribution. At the same time, they applied Image Restoration on the input image prior to assessing the residual image which proved to give significant performance improvements. [Baur et al., 2020] presented a comprehensive comparative study for recent approaches to unsupervised lesion detection. Fig. **??** shows input with ground-truth annotations and the corresponding reconstructions and residual maps for different models employed in their work.

While the approaches mentioned before are based on auto encoder frameworks, there are interesting works inspired by the GAN framework. [Schlegl et al., 2017, Sch, 2019] have developed the AnoGAN and f-AnoGAN frameworks, the latter of which showed competitive performance with VAE (with restoration applied) based models in the comparative study above. However, unlikely VAE-based models, GANs do not offer a way to encode an input sample into the latent space due to which an optimization problem has to be achieved to infer the latent space representation before the decoder can produce a reconstruction.

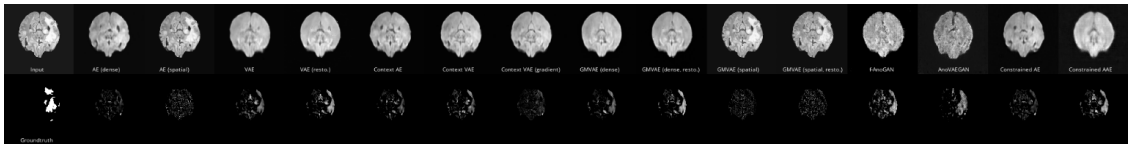


Figure 3.6: [Baur et al., 2020] compared a wealth of different unsupervised anomaly detection frameworks and reported their relative performance in a standardized experiment on.
Source: [Baur et al., 2020]

3.4 Supervised Lesion Detection

As introduced in Sec. 2.1, supervised approaches are free to use ground-truth labels during training. In the field of medical image segmentation, supervised models are often based on or variants of the infamous U-Net architecture [Ronneberger et al., 2015]. As a matter of fact, the winning submission for the 2020 Brain Tumor Segmentation Challenge (BraTS) is such a variant [Isensee et al., 2020], using a cascaded U-net architecture. While supervised lesion detection is an extensive topic in itself, it is not very relevant for this work and will therefore not be discussed in great detail.

3.5 Learning Disentangled Representations

Much interest has been put into learning meaningful representations with disentangled influencing factors. In the case of a VAE this means that it would be desirable to have single dimensions of some latent space representation encode isolated hidden factors. This could be, for example, the size of a brain, some notion of shape or the level of diseasedness. [Higgins et al., 2017] have shown that by simply increasing the weight (called β) of the KL-divergence term in equ. 3.4 from 1.0, which would correspond the original VAE framework, to something larger, one is able to learn a more disentangled latent space representation. That means that a certain type of variation should be encoded in a single dimension when traversing the latent space. The main idea is rather simple: When enforcing a multivariate, isotropic Gaussian prior on the latent space, then, due to it being isotropic, it's components are independent of each other, or in other words, disentangled. The stronger the latent space representation follows this prior, the stronger the disentanglement. Hence, by introducing a simple modification to the VAE cost function, namely up-weighting the KL-term one can achieve better disentanglement of the individual underlying factors. By doing so, the regularization on the latent space is increased and it gets less flexible while at the same time the KL term starts become more important compared to the reconstruction loss which results in blurrier reconstructions overall.

Figure 3.7: β-VAE ($\beta = 250$) achieves to disentangle underlying factors such as azimuthal rotation much better than the standard VAE ($\beta = 1$) implementation at the expense of resolution. Note that the standard VAE is able to learn notions of azimuthal angles as well but entangles the representation with other factors such as lightning or hair stlye.
Source: [Burgess et al., 2018]

Chapter 4

Materials and Methods

4.1 Unsupervised Lesion Detection using VAE's

4.1.1 Working Principle

This section shall explain in more detail the concepts of unsupervised lesion detection based on VAEs, the core objective of the models used throughout this work.

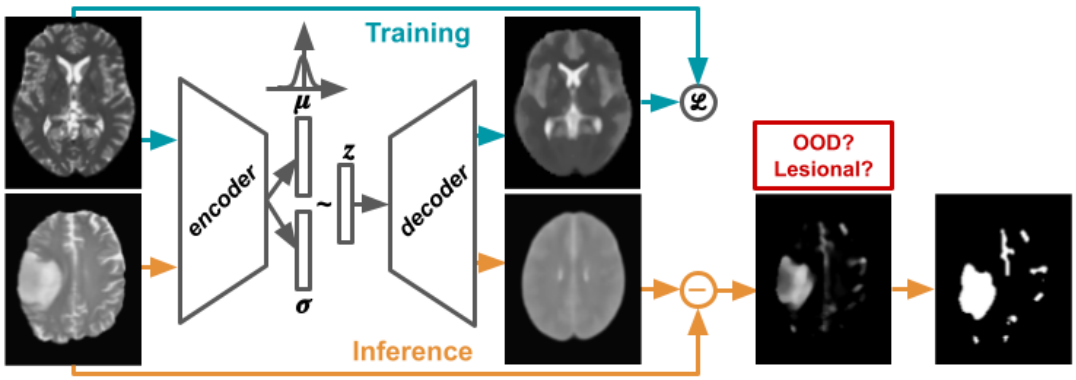


Figure 4.1: Working principle of unsupervised lesion detection based on VAEs as used in this work. During training, a VAE learns to approximate the distribution of healthy images by maximizing the so-called Evidence Lower Bound (ELBO) $\mathcal{L}$ (cf. Equ.3.4). During inference, the residual map $\mathbf{r}$ of an input test image yields the pixel-wise lesion detection map. Finally, the resulting binary pixel-wise lesion prediction is obtained by comparing every residual pixel with a threshold τ which has been determined using training data only. Unlike previous works, this framework aims to answer the question due to which underlying factors a sample is being predicted to be OOD.

Using a VAE trained on a set of healthy samples, a potentially lesional test sample $\tilde{\mathbf{x}}^{(i)}$ is being fed through the model to obtain a reconstruction $\hat{\mathbf{x}}^{(i)}$. Being trained on healthy samples exclusively, the model is expected to not being able to reconstruct lesional components while it should succeed in reconstructing healthy parts of the input. A pixel-wise anomaly segmentation map can thus be retrieved by thresholding the pixel-wise residual $\mathbf{r} = \|\hat{\mathbf{x}}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|_p \in \mathbb{R}^{m \times n}$, where $\|\cdot\|_p$ is the ℓ_p-norm, and p is chosen to be 1.

A pixel is being marked as anomaly if the per-pixel value in the residual image $\mathbf{r}$ exceeds some threshold τ. Since, in an unsupervised setting, there is no access to ground truth labels, it is not possible to tune such hyper-parameters with regards to the final metrics of interest, e.g. the Dice score as one would typically do in a supervised setting. Instead, we follow the approach in

[Konukoglu and Glocker, 2018], where we assume any pixel from the training set images marked to be anomalous by the anomaly detection algorithm to be a false positive. We set a limit l_{FPR} (typically 5%) on the false positive rate FPR_{train} we are willing to accept to determine a threshold that satisfies the constraint. The threshold is computed via the Golden Section Search algorithm [Kiefer, 1953] by performing the optimization problem

$$\tau = \min_t |FPR_{train}(t) - l_{FPR}| \tag{4.1}$$

Finally, the threshold τ gets deployed to convert the residual map $\mathbf{r}$ to a binary lesion map for which the final segmentation metrics are being computed per patient on unseen test data.

4.1.2 Baselines

For **pixel-wise lesion detection** we compare to the baseline provided by [Chen et al., 2020] which denotes the current state-of-the-art performance in unsupervised lesion detection on brain MRI images. They use the T2-weighted images from the CamCAN dataset for training and test on T2-weighted images from the BraTS 2017 datasets, similar to our experiments.

Model	$Dice$	AU_{ROC}
VAE128	0.23 ± 0.13	0.69
GMVAE	0.46 ± 0.23	0.83

Table 4.1: Baseline performance for pixel-wise lesion detection from [Chen et al., 2020]. While the $VAE128$'s model architecture is comparable to ours, $GMVAE$ uses a Gaussian mixture model to model the latent space distribution and deploy restoration of the input image which has been shown to yield significant performance boosts by [Baur et al., 2020]. For the $Dice$ score, the patient-wise mean and standard deviation is reported.

Slice-wise or sample-wise lesion detection is done significantly less frequently in the field of unsupervised lesion detection. However, recently [Zimmerer et al., 2020] performed slice-wise lesion detection using both terms from Equ. 3.4 as well as the full ELBO $\mathcal{L}$. They train on the HCP [Van Essen et al., 2013] dataset and test their model on the BraTS 2017 challenge dataset as well. While no exact numbers are given, their plots suggest an AU_{ROC} of about 0.70 to 0.75 for a latent-space dimension of 128. They found that the KL-term performed just slightly superior compared to the reconstruction error or full ELBO.

Model	AU_{ROC}
VAE	0.70 to 0.75

Table 4.2: Baseline performance for slice-wise lesion detection from [Zimmerer et al., 2020].

To the best of our knowledge there are no works applying recent **OOD detection** techniques in the field of unsupervised lesion detection on brain MRI images which is why there exists no baseline to compare to.

4.2 Possible Evaluation Metrics

This section provides an overview and explanation as to which metrics have been used to evaluate our experiments and the rational behind that.

In the setting of pixel-wise lesion detection, we can test either on a set of healthy images, lesional images or a combination thereof. Note that when using healthy images only in the test set,

that is images which contain no lesions, there are no actual positives, as shown in Fig. 4.2. Here, we denote *actual positives* as pixels which are actual lesional based on ground truth annotations. Hence we can define the ROC and PRC curves only when testing on lesional slices only or a combination of healthy and lesional slices as for example the raw BraTS 2017 dataset. Reporting the ROC and PRC curves in the case of slice-wise lesion detection is only possible when considering a test set containing both healthy and lesional slices.

Pixel-wise Anomaly Detection		Test healthy	Test lesional	Test combo
Actual Positives	**True Positives** is lesional - predicted lesional	✗	✓	✓
	False Negatives is lesional - predicted healthy	✗	✓	✓
Actual Negatives	**False Positives** is healthy - predicted lesional	✓	✓	✓
	True Negatives is healthy - predicted healthy	✓	✓	✓

	Test healthy	Test lesional	Test combo
ROC TPR - FPR	recall not defined	✓	✓
PRC PPV - TPR	precision always = 0 & recall not defined	✓	✓

Slice-wise Anomaly Detection		Test healthy	Test lesional	Test combo
Actual Positives	**True Positives** is lesional - predicted lesional	✗	✓	✓
	False Negatives is lesional - predicted healthy	✗	✓	✓
Actual Negatives	**False Positives** is healthy - predicted lesional	✓	✗	✓
	True Negatives is healthy - predicted healthy	✓	✗	✓

	Test healthy	Test lesional	Test combo
ROC TPR - FPR	recall not defined	no actual negative	✓
PRC PPV - TPR	precision always = 0 & recall not defined	precision always = 1	✓

Figure 4.2: Overview on as to which metrics can be used given the classification scenarios for pixel-wise and slice-wise anomaly detection.

For OOD detection, we actually don't care if we test on healthy or lesional slices when it comes to metrics since the ground-truth annotation in this case has nothing to do with the presence or lesions but is concerned with labelling samples being in- or out-of-distribution. Therefore we have the freedom to split up the analysis of OOD detection into the three groups, that is healthy samples, lesional samples or a combination thereof, as shown in Fig. 4.3.

		OOD Dataset		
OOD Detection		Test healthy	Test lesional	Test combo
Actual Positives	True Positives is OOD - predicted OOD	✓	✓	✓
	False Negatives is ODD - predicted ID	✓	✓	✓
Actual Negatives	False Positives is ID - predicted OOD	✓	✓	✓
	True Negatives is ID- predicted OOD	✓	✓	✓

	Test healthy	Test lesional	Test combo
ROC TPR - FPR	✓	✓	✓
PRC PPV - TPR	✓	✓	✓

Figure 4.3: Overview on as to which metrics can be used given the classification scenario of OOD Detection.

4.3 Conditioning the Latent Space

Sec. 5.5 introduced the trade-off between fidelity of reconstructions and a stronger regularization of the latent space by weighting the KL-divergence term of the VAE loss function (cf. Equ. 3.4). For small values of β, the reconstruction error has a higher relative weight which leads to sharper reconstructions. At the same time there is less regularization on the approximate posterior which means more entanglement of latent space dimensions. Also, there is more freedom for the learned approximate posterior to deviate from the imposed prior, which in our case is a multivariate Gaussian with zero mean and unit variance. When lowering β too far, we approach the limiting case of a standard Auto Encoder and end up with a less expressive latent space. There, it should not be possible to sample from the prior distribution and let the decoder produce samples following the training distribution anymore. Therefore, it is desirable to find a β sweet-spot where

- high-quality reconstructions are possible (lower β better),
- it is possible to sample from the prior and receive reconstructions following the training distribution for an interpretable latent space behaviour (higher β better),
- the approximate posterior is not ignored (lower β better).

The last point of the above list is what is referred to as *posterior collapse* and reflects the issue that the approximate posterior collapses $q(\mathbf{z}|\mathbf{x})$ to the imposed prior $p(\mathbf{z})$, i.e. $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$. The model essentially learns to ignore the latent variable. Understanding the causes of posterior collapse and its implications is an active area of research but recent work has shown that it is likely to be connected with training dynamics [He et al., 2019].

The experiments to fulfill the above requirements are shown in Sec. 5.4. They are essential to continue in analysing the potential of learning disentangled representations.

4.4 What Can Pre- and Post-processing do for us?

Histogram Matching [Chen et al., 2020] used histogram matching (cf. Sec. 5.2) to improve their lesion segmentation performance. However, there was no indication given as how much improvement can be achieved by doing so. So a first natural experiment is to quantify this effect, the results of which are shown in Tab. 5.1. Furthermore, it is expected that histogram matching reduces domain-shift effects which is important to realize since this in turn effects the capability to detect lesional samples. This is an important insight since one might argue that histogram matching could potentially solve the problem. We show that it does affect sample-wise anomaly detection only mildly by comparing loss term histograms in Fig. 5.15.

Post-processing the residual map Tab. 5.1 also shows the quantitative effects of applying the post-processing effects explained in Sec. 5.2. It should be noted that these techniques might introduce prior knowledge deviating away from the idea of a completely unsupervised system. Nevertheless, the processing steps applied are mild and assume very little knowledge about the nature of lesions but rather represent sensible image processing techniques to reduce noise.

4.5 Disentanglement of Lesions and Domain-shift Effects

At test time, the model might be exposed to samples from the following three categories (cf. Fig. 5.14 which considers CamCAN T2 to be in-distribution and BraTS T2 healthy and BraTS T2 lesional OOD):

1. **healthy & in-distribution**, anomaly-free images from the same domain as the training data (e.g. CamCAN T2)
2. **healthy & OOD**, anomaly free-images with domain shift from the training data (e.g. BraTS T2 healthy),
3. **lesional & OOD**, images with lesions regard less of domain shift (e.g. BraTS T2 lesional)

.

A practitioner is interested in a model which answers the question "Does this sample contain lesions?". He is not interested in the question "Has this sample undergone a domain shift?". Rather he would like to have the additional question "Can I trust this prediction?" answered. Note that for our case the last two questions actually have the same meaning. When deploying an unsupervised lesion detection model, a high anomaly detection score, for example the reconstruction error, might tell him something like "This sample could be lesional. Or, it could be perfectly healthy but is domain shifted, that is, it is very different from the samples I have seen during training. I cannot tell you what drives the anomaly score though.". This is reflected by the experiments and results presented in Sec. 5.7.

To enable save deployment in practice, one would have to be either absolutely sure the model is only fed with samples originating from the same domain or learn to differentiate between the two sources of error. Since the former is difficult to achieve and maintain in practice and does not give any insights from a research point of view, we discuss how to potentially achieve such disentanglement up next.

4.5.1 Learning Disentangled Representations via β-VAE

Learning disentangled representations via the β-VAE framework has been introduced in Sec. 3.5. We conduct several experiments to find out whether this can help in disentangling the underlying factors for a sample to have high reconstruction error. First, the baseline model is properly conditioned as described before. Then, we assess the effect a stronger latent space regularization has on the learned representation.

The corresponding experiments and results can be found in Sec. 5.5 while further insights can be found in the appendix in Sec. A.2.

4.5.2 The Entropy Score

Our experiments (cf. Sec. 5.7) indicated that when performing unsupervised lesion detection, errors originating from domain-shift effects are inherently intertwined with such originating from lesions themselves. The entropy score proposed in the following is an opinionated approach to tackle

Assumption Lesional samples are expected to have large residual errors confined in relatively small regions considering pixels within a tumor dominating the reconstruction error. On the other hand, a (healthy) OOD sample should show a steady but spread out error due to global domain-shift effects which is equivalent to a large uncertainty in the pixel-wise error distribution which might be captured by the following entropy measure. We extend the framework with an entropy statistic H_{ℓ_1} and investigate its suitability to disentangle the underlying reasons for which a sample appears to be marked OOD. The normalized sample-wise entropy scores are calculated on a normalized (sums up to 1) residual map $\mathbf{r}$ via

$$H_{\ell_1}(\mathbf{r}) = -\sum_{j}^{n_p} \frac{\mathbf{r}_j \log \mathbf{r}_j}{\log n_p}. \tag{4.2}$$

where n_p is the number of pixels within the brain mask. Note that by doing so, we implicitly assumed the normalized residual image to be a discrete n_p-dimensional probability distribution. Furthermore we should note that the entropy score does not take spacial information into account.

4.5.3 Recent OOD metrics

We have described earlier and backed up the hypothesis by experiments (cf. Sec. 5.5) that unsupervised lesion detection is inherently equated or even based on OOD detection. Thus, it stands to reason to assess their usefulness in sample-wise lesion detection. A metric which performs well in this experiment can be expected to be more susceptible to effects originating from lesions which make the sample deviate from the training distribution. The associated results are listed in Tab. 5.2.

4.6 Implementation Details

The major model used to carry out the experiments throughout this work is a VAE following the architecture in [Baur et al., 2020] while also the model used by [Zimmerer et al., 2020] has been tested without significant discrepancies in performance the the former whose model architecture is being depicted in Fig. 4.4. Each layer consists of a 2D convolution (partially transpose convolutions in the decoder) followed by batch normalisation and a LeakyReLU activation function. Models are trained for 80 epochs with a batch size of 128. Different β-annealing schedules have been tested (cf. Sec. A.3) while, if not stated otherwise, monotonic β-annealing from 0 to 0.4 over the first 2000 training steps is applied. Adam optimizer [Kingma and Ba, 2017] has been used with an initial learning rate of 10^{-4}.

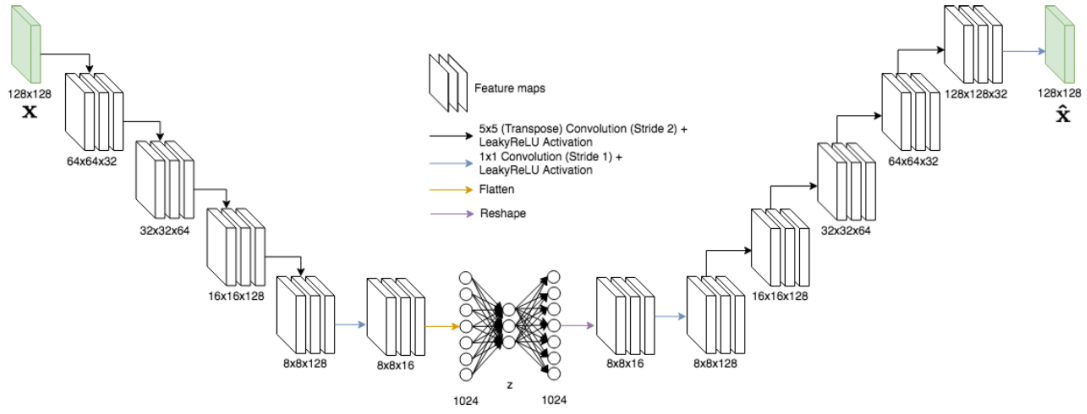


Figure 4.4: The VAE architecture used throughout the experiments of this work.
Source: [Baur et al., 2020]

4.7 Software & Hardware

Software This project has been carried out using the *PyTorch* [1] deep learning framework, arguably the most used framework in machine learning research community. To improve reproducibility and ease of development, *Pytorch-Lightning* [2] was used as an additional abstraction

[1] https://pytorch.org/
[2] https://github.com/PyTorchLightning/pytorch-lightning

layer on top of PyTorch. To track ongoing experiments, *Tensorboard* [3] has been used. Additional dependencies included *h5py* [4] to create and manage datasets during training and inference, *OpenCV* [5] for image processing and *nibabel* [6] to process neuro-imaging file formats. To ensure reproducibility of the results, every experiment on the modular evaluation pipeline is connected with logged hyperparameters which get stored along the results. The project was developed from the ground up resulting in around $13'000$ lines of *python* [7] code and is publicly available at `https://github.com/matthaeusheer/uncertify`.

Hardware The workstation used for training and evaluation included a 6 core Intel(R) Core(TM) i7-8700 CPU running at a frequency of 3.20GHz with 16GB RAM and a NVIDIA GeForce GTX 1060 graphics card with 6GB VRAM. Training a single model usually took around 5 hours.

[3] https://www.tensorflow.org/tensorboard
[4] https://www.h5py.org/
[5] https://opencv.org/
[6] https://nipy.org/nibabel/
[7] https://www.python.org/

Chapter 5

Experiments and Results

5.1 Datasets

The datasets used in this work are subdivided into a group of MRI based brain scan datasets for which samples are shown in Fig. 5.1 and a group of general purpose toy datasets shown in Fig. 5.2. Datasets from the first group are essentially variants of the CamCAN [Taylor et al., 2017] and *Multimodal Brain Tumor Segmentation Challenge 2017* (BraTS) [Menze et al., 2014] datasets. The second group is comprised of datasets oftenly used in machine learning research to enable fair comparisons and proof of concepts. Whenever a dataset is used for training, a 90%-10% train-test split is applied. Images are cropped and reshaped to obtain a resolution of 128×128 pixels. In the experiments shown, only CamCAN T2 is used for training while variants of the BraTS dataset are used for testing only, so no split has been applied there.

The following datasets are all based on MRI data. Throughout this work we used T2- and T1-weighted images exclusively with several modifications applied during the pre-processing stage which are explained next.

- **CamCAN T2**
  If not stated otherwise, models used throughout this work have been trained on T2-weighted images from the CamCAN dataset. It consists of a total of full brain scans from 653 patients. The histograms from all samples from this dataset have been matched patient-wise against a randomly chosen patient from the training split to ensure common histogram distributions throughout the training dataset.

- **CamCAN T2 lesion**
  This is an adaption to the CamCAN T2 dataset in which Gaussian blobs have been pasted on top of the slices at random as described in Sec. 5.2.

- **BraTS T2**
  We merged different categories of severity classes of labels into one to obtain a binary lesion map. Our dataset consists of 75 scans from low-grade glioma patients and 210 scans of high-grade glioma patients. There are no healthy patients contained in this dataset.

- **BraTS T2 HM**
  Similar to BraTS T2 but with histogram matching against the very patient histogram distribution against which all training samples have been matched against.

- **BraTS T1**
  T1-weighted images from the BraTS 2017 dataset.

- **BraTS T1 HM**
  Similar to BraTS T1 but with histogram matching applied similar to BraTS T2 HM.

- **BraTS T2 HFlip**
  Horizontally flipped version of BraTS T2. Those flipped datasets are common to test against in the OOD detection community, since they can be regarded as OOD while still being close to the original dataset.

- **BraTS T2 VFlip**

  Vertically flipped version of BraTS T2. Since brain scans are more or less horizontally symmetric, a vertical flip is a much more severy transformation compared to a horizontal flip.

- **IBSR T1**
  For this dataset contains only T1-weighted images. It holds scans from 20 healthy subjects and has been used for auxiliary tests throughout development.

- **CANDI T1**
  This dataset contains only T1-weightd images. It has been used on auxiliary tests not reflected in the reported experiments.

The following datasets are not based on MRI images. They are clearly OOD for models trained on MRI data and can be used to test simple ideas or be used in OOD detection experiments.

- **MNIST**
  The MNIST dataset [LeCun and Cortes, 2010], first introduced in 1998, consists of 70́00 greyscale images of handdrawn digits. It has been used in a wealth of works throughout the years and is arguably the most famous dataset in the machine learning community.

- **FashionMNIST**
  Since the MNIST dataset has become arguably to easy for most modern deep learning methods, the FashionMNIST [Xiao et al., 2017] has been created to pose greater challenges to image classifiers and other models. It consists simple greyscale depictions of clothings.

- **Gaussian Noise**
  This dataset consists of images with random Gaussian noise using mean zero and standard deviation one.

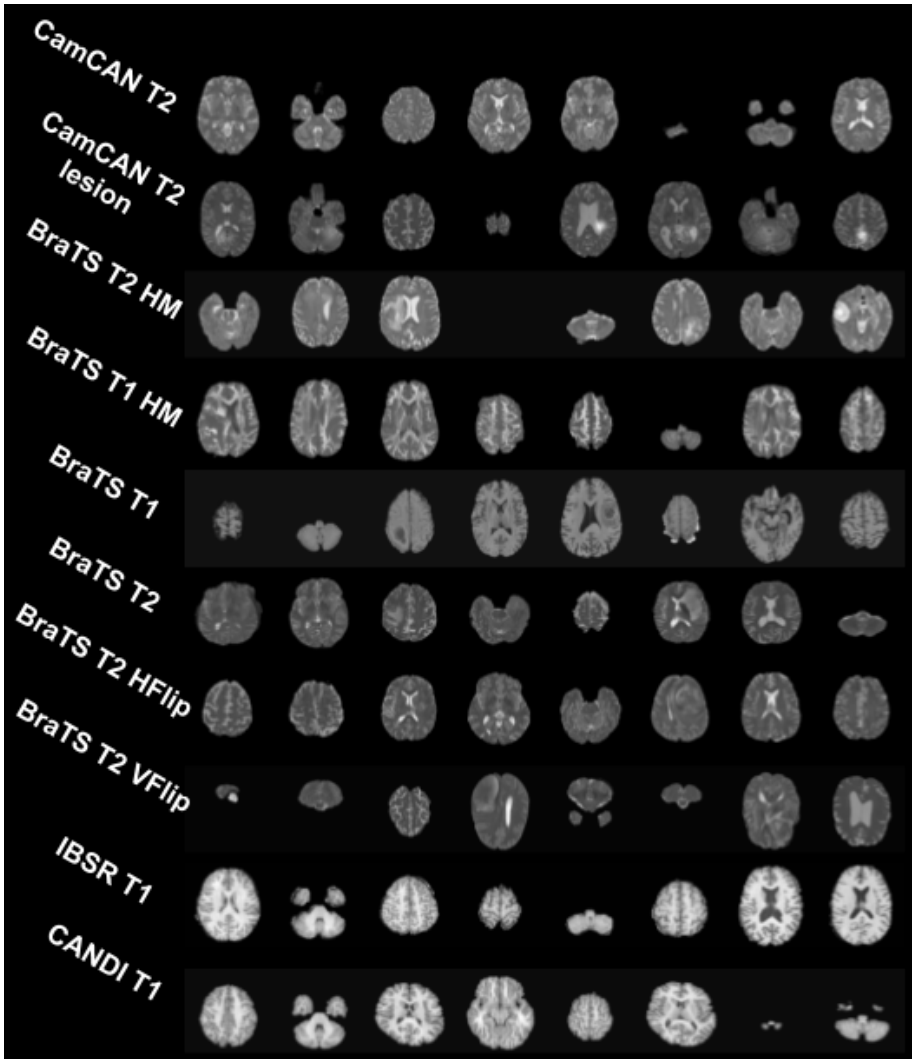


Figure 5.1: Visual Examples of MRI based datasets.

Figure 5.2: Visual Examples of toy datasets used primarily in OOD detection experiments.

5.2 Pre- & Postprocessing

Pre-processing For MRI-based data, pre-processing includes bias correction using the N4 algorithm [Tustison et al., 2010] and centering of the brain. Bias correction aims to correct low-frequency non-uniformities which are commonly seen in MRI data and are known as the bias field. Patient-wise histogram matching to a randomly chosen in-distribution sample ensures similar intensity profiles throughout all training samples. Fig. 5.3 shows at an example of two patients how their intensity profiles change and start to match the reference profile more closely.

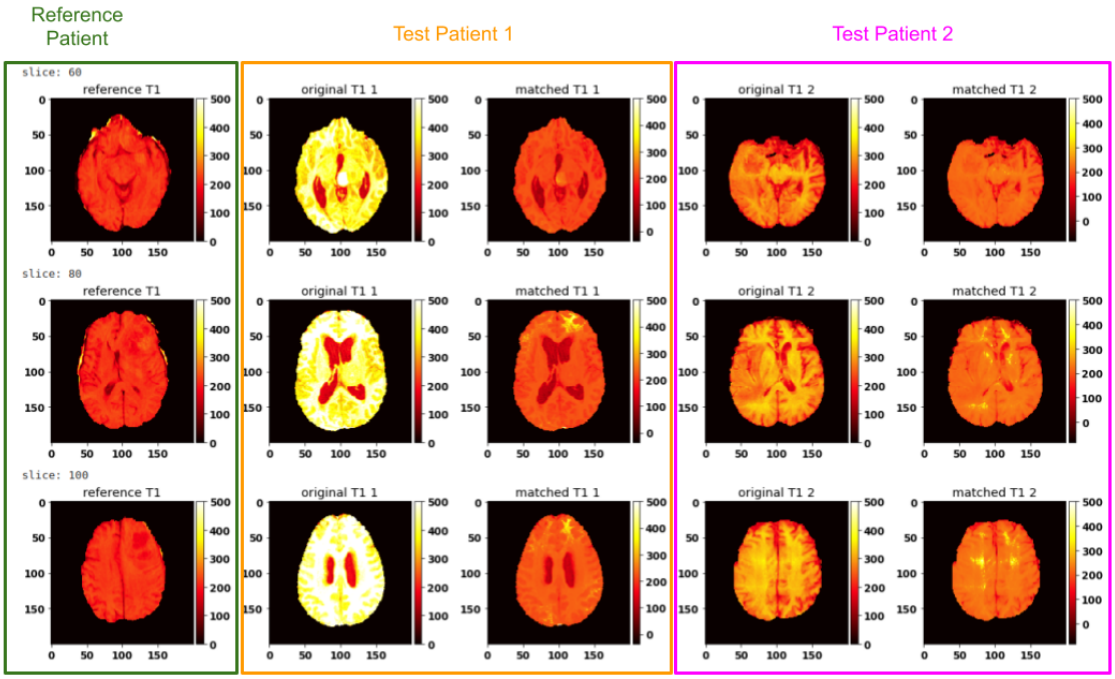


Figure 5.3: The patient-wise histograms of two test patients are being matched against the patient-wise intensity histogram of a reference patient.

Finally, all pixel values within the brain mask are normalized to zero mean and unit variance, again, on a per-patient level. Image sizes are cropped to 128×128 pixels. Resulting samples are shown in Fig. 5.4.

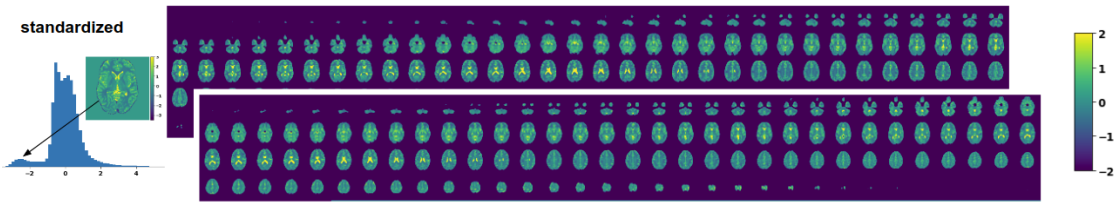


Figure 5.4: All slices for two patients from the CamCAN T2 dataset after standardization, that is, subtracting slices the mean and dividing by the standard deviation both of which have been calculated over all pixels from a single patient beforehand. The histogram on the left shows intensity values from a single patient. The bump on the left side can be explained due to imperfections in the brain mask which adds low-valued background pixels to the distribution.

Empty MR slices are excluded during training. To obtain lesional samples following the same distribution as CamCAN T2, we artificially crafted lesional samples by randomly adding high-intensity Gaussian blobs to CamCAN T2-weighted images. The blobs standard deviations range

from 0 to 10 (given images of size 128×128) and are weighted such as to have similar maximum intensities compared to real lesions. Finally, we applied vertical and horizontal flipping (V-flip, H-flip) following common practices in recent OOD detection works.

Post-processing The following post-processing steps have been implemented and evaluated. These techniques have been introduced by [Baur et al., 2020] and apply prior knowledge to common failure modes and the nature of how lesions show up in MRI images, potentially limiting its usability with the benefit of higher lesion segmentation performance. The (significant) effect of the post-processing steps described below is shown in Fig. 5.5.

1. **The l_1^{max}-residual** Lesions contained in BraTS (glioma) appear bright in a T2-weighted image. For that reason the l_1-residual between the input and reconstructed image is usually positive for lesional matter. Thus, all residual pixels smaller than zero can be set to zero.

2. **Median Filtering** OODs are in general not spoiled for being able to reconstruct high-frequency content which yields reconstructions to appear blurry. On the other side, high-frequency content appears as small patches in the residual map without belonging to an actual lesion. By applying a median filter to the residual map, this issue can be mitigated to some extent.

3. **Brain Mask Erosion** A common failure mode of models described in this work is that of highly erroneous regions at the brain boundary. These pixels usually contribute to the set of false positives which is why one might want to erode the brain mask inwards on the residual map to get rid of those wrongly reconstructed pixels.

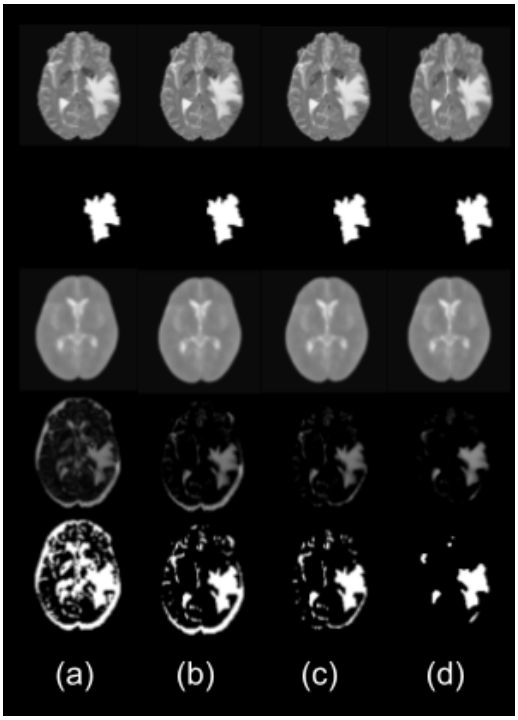


Figure 5.5: Rows from top to bottom: Original image, ground-truth tumor segmentation, reconstruction, residual image, binary tumor prediction map. Columns: (a) No post-processing applied. (b) Instead of the raw l_1 distance from input to reconstruction, set pixels to zero which are brighter in the output image compared to the input. (c) Apply brain-mask erosion to get rid of boundary effects. (d) Apply median filtering to smooth out the residual map. Note that the threshold for this image has been found with all post-processing enabled. Nevertheless, there one cannot obtain the same quality of segmentation as in column (d) without the processing steps.

5.3 Determining the residual threshold

As described in Sec. 4.1, the threshold on the residual map is determined on training data only by solving the optimization problem of Equ. 4.1. Fig. 5.6 shows this procedure for a model trained on CamCAN T2. In this case, post-processing to the residual map in form of brain mask erosion and median filtering has been applied which reduces the number of false positives considerable and yields a threshold $\tau = 0.70$. When not applying those post-processing steps the threshold found is significantly higher at around $\tau = 1.50$.

In the 5^{th} column of Fig. 5.7, one can observe a common failure mode of VAEs, which is the inability to reconstruct high-frequency content faithfully due to the characteristic blurriness of reconstructions.

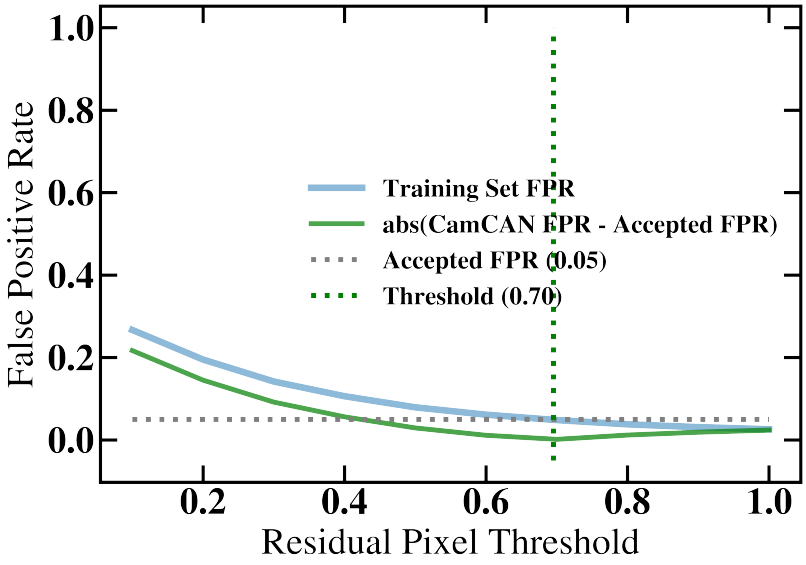


Figure 5.6: Optimization procedure to determine the threshold τ on the residual map $\mathbf{r}$ by setting an accepted False Positive Rate (FPR) on the training data (5%) and searching for a threshold which produces a FPR on the training set which is closest to this value. The optimization is carried out using the Golden Section Search algorithm.

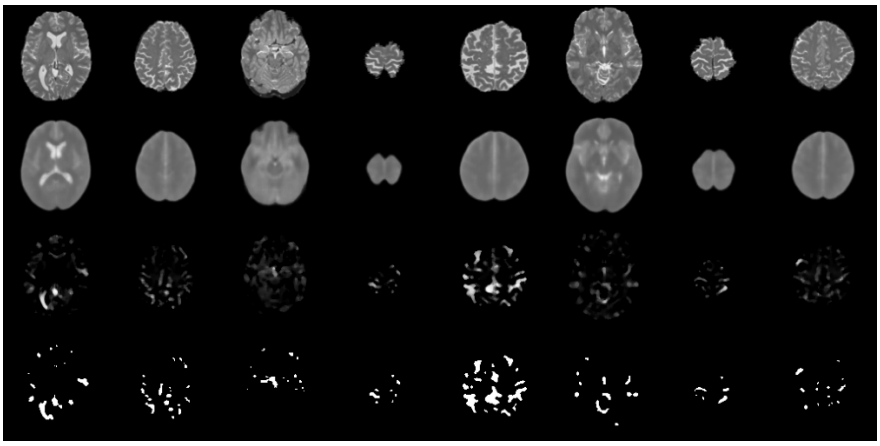


Figure 5.7: From top row to bottom: Random input samples from CamCAN T2, associated reconstructions, l_1 residuals and resulting binary pixel maps when applying a threshold of $\tau = 0.7$ which results in a false positive rate of max. 5% throughout the dataset.

5.4 Conditioning the Latent Space

To correctly condition the latent space of the VAE, a search is performed to find the optimal β value as described in Sec. 4.3.

Fig. 5.8 shows decoded reconstructions when sampling from a Gaussian prior distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. While models trained with low values of β do not yield realistic reconstructions, there is a sweet-spot at $\beta = 0.5$ where realistic reconstructions result from decoded latent space samples. Higher values of β loose fidelity in reconstructions and are more prone to result in what we believe is a manifestation of posterior collapse (see Fig. 5.9).

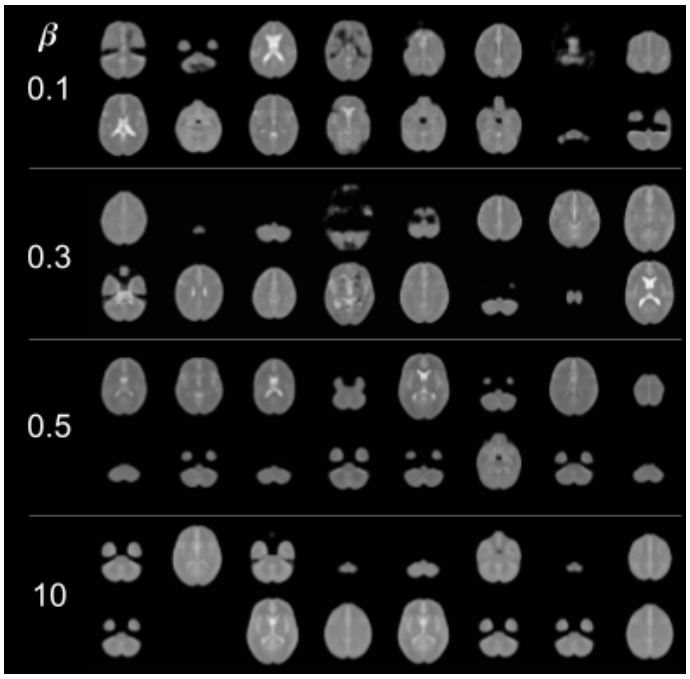


Figure 5.8: Reconstructions of latent-space samples which follow a multivariate Gaussian with zero mean and unit variance. Whe the model has been trained with very values of β (0.1 - 0.3), the approximate prior is not regularized enough and the decoding process will not yield realistic reconstruction. For higher values of β, the approximate prior mimics the Gaussian prior more closely and realistic reconstructions are obtained. For very high values of β (e.g. 10), a loss of resolution can be observed.

Fig. 5.9 reflects what we believe is a manifestation of posterior collapse, that is, no matter what the input to the model is, it learns to bypass the latent space representation and reconstructs samples from the distribution it has been trained on. As a matter of fact, this puzzling behaviour has been first observed due to a bug in the loss function implementation where the actual loss was heavily dominated by the KL divergence term which is essentially the same as choosing a high β.

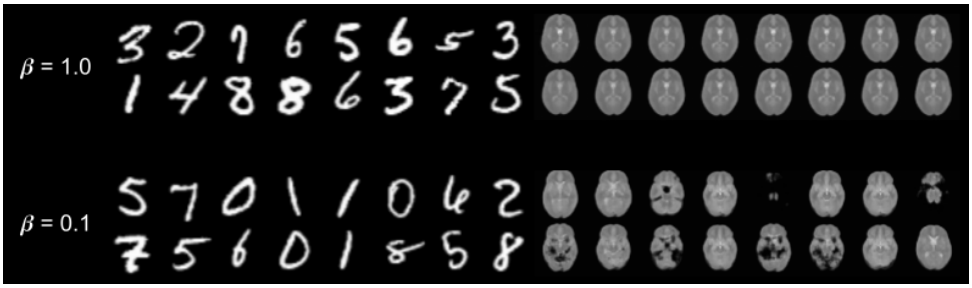


Figure 5.9: Reconstructions of MNIST input images. Left: MNIST input (batch of 16 samples each), right: corresponding reconstructions. The top model has been trained with $\beta = 1.0$ (original VAE objective) while the bottom model has been trained with a lower latent space regularization of $\beta = 0.1$. We believe that the top model suffers from posterior collapse while the reconstructions of the bottom model indicate a proper latent space conditioning due to the fact that it is not able to reconstruct the heavy OOD input samples but is still able to reconstruct MRI brain images faithfully.

5.5 Learning Disentangled Representations using β-VAE

The following experiment reveals the interesting fact that some in-distribution input samples cannot be reconstructed faithfully when choosing a high β value. The original work [Higgins et al., 2017] on β-VAE proposes β values up to 250. A follow-up work aiming to explain the working principles of the technique more closely works with a value of 150. In those works they usually show reconstructions from samples obtained by traversing the latent space, so the particular issue that some in-distribution input samples can not be faithfully reconstructed as shown in Fig. 5.10 has not been observed before by us. However, this behaviour is probably not surprising in the sense that the reconstruction term is not important enough in the objective function to obtain faithful reconstructions. Nevertheless, since the framework of unsupervised lesion detection depends on truthful reconstructions of healthy parts of the brain, we can conclude that such a model can not be deployed for reconstruction-based anomaly detection, even though it might be able to learn an expressive latent space representation.

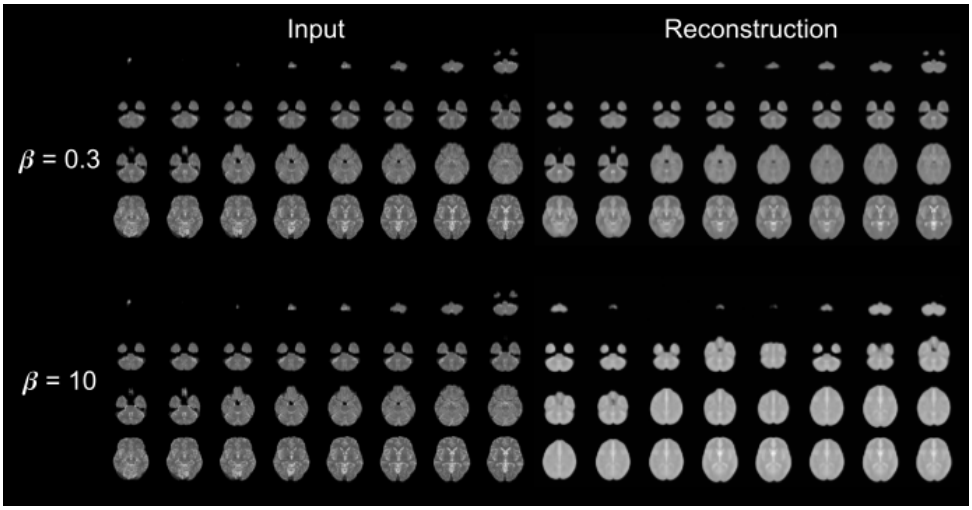


Figure 5.10: Comparing reconstructions from two models trained with different KL-term weights (0.3 and 10). The model with a high weight is not able to reconstruct all input samples faithfully rendering it unusable for unsupervised lesion detection.

Future work in this direction might consider investigating the following algorithm which we propose based on our empirical findings.

1. Train a model *A* with a "sweet-spot" β, in our case $\beta = 0.5$ as explained before. This model will eventually be used to perform unsupervised lesion detection.

2. Train a second model *B* with a high value of β, polluted with data from different domains. It has to be verified that this model is actually able to disentangle factors inherent to any MRI image dataset, such as shape and size of brains, with factors which reflect domain shifts such as noise behaviour and intensity changes.

3. Finally, one could use the latent dimensions associated with domain shift factors as a domain-shift detection metric. One would perform lesion detection inference on model *A* only if the latent space of model *B* has shown that there has no domain shift been detected.

While this seems to be an interesting line of work and several follow-up works [Kim and Mnih, 2019, Chen et al., 2019] have attempted to improve the basic β-VAE framework, we leave the exploration of these frameworks for the task at hand for future work. Additionally, it should be pointed out that there is a GAN-based line of work which attempts to learn disentangled representations [Chen et al., 2016] as well. Instead, we explore techniques based on current advances in the field of OOD detection for generative models in order to disentangle lesions and domain-shift influences directly.

Sec. A.2 gives more insights into the behaviour of a model with high β to learn disentangled factors of variation in the data.

5.6 Pixel-wise anomaly detection

Tab. 5.1 summarizes the pixel-wise lesion detection from our model with and without the post-processing introduced in Sec. 5.2 applied. While achieving high segmentation performance is not the main objective of this work, this analysis situates our model and approach compared to the state-of-the-art. Without post-processing the model performs on par compared to the baseline while post-processing (eroded brain mask and median filtering on the residual map) brings a 26.5% increase in Dice score. Nevertheless, the GMVAE approach including image restoration still performs significantly better. Histogram matching brings a 9% performance (Dice) in crease when post-processing is applied and 56% increase without post-processing.

Table 5.1: Pixel-wise anomaly segmentation (Dice) and detection (AU_{ROC} / AU_{PRC}) performance. * includes post-processing (smoothing & mask-erosion of residual map) , ** no post-processing, *** results from [Chen et al., 2020]. Dashed entries were not provided by the authors.

Dataset	BraTS T2 HM			BraTS T2			CamCAN T2 artificial lesions			BraTS T1		
Model	Dice	AU_{ROC}	AU_{PRC}	Dice	AU_{ROC}	AU_{PRC}	Dice	AU_{ROC}	AU_{PRC}	Dice	AU_{ROC}	AU_{PRC}
Baseline***	0.23 ± 0.13	0.69	-	-	-	-	-	-	-	-	-	-
Ours VAE*	0.34 ± 0.12	0.75	0.25	0.31	0.73	0.20	0.63	0.92	0.66	0.10	0.51	0.07
Ours VAE**	0.25 ± 0.11	0.69	0.16	0.22	0.66	0.13	0.39	0.88	0.43	0.10	0.49	0.07
GMVAE***	0.46 ± 0.23	0.83	-	-	-	-	-	-	-	-	-	-

Artificial lesions Lesion detection performance on the artificially crafted lesional CamCAN T2 dataset is significantly higher compared to running inference on BraTS T2. A sample batch prediction for running inference on this dataset is shown in Fig. 5.11. This reflects the scenario when one would test on lesional images which, lesions set aside, originate from the same data distribution compared to the training images. The performance improvement is significant moving from

a Dice score of 0.34 (BraTS T2) to 0.63 (CamCAN T2 lesion). Note that the artificially pasted lesions differ strongly from actual lesions which is why such an experiment should be taken with a grain of salt. On the other hand, it serves as a confirmation that domain-shift effects can not be ignored and as a motivation to investigate further on this issue in the context of unsupervised lesion detection.

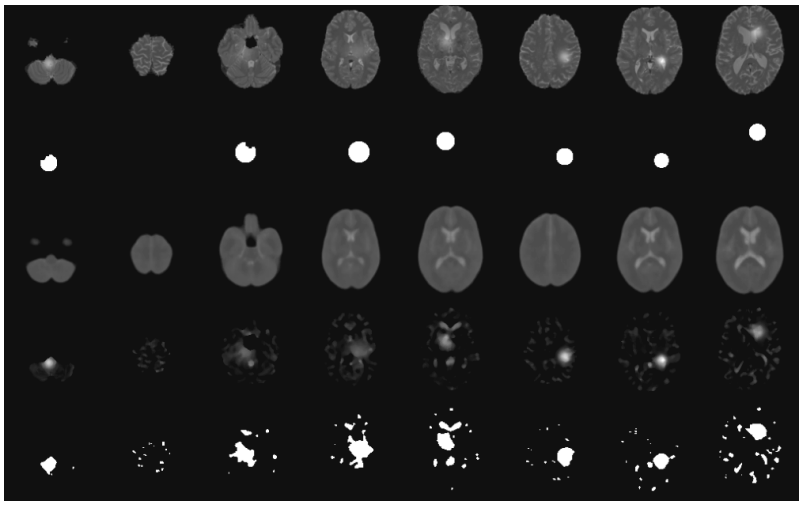


Figure 5.11: Running inference on samples from the CamCAN T2 lesion dataset, which contains artificially crafted lesions in form of Gaussian blobs. Rows from top to bottom: Random input samples, ground truth segmentation obtained by the 1σ region around the mean of the Gaussian blob, reconstruction, residual, binary prediction.

Fig. 5.12 shows reconstructions and associated predictions when running inference on BraTS T2 for a model trained on CamCAN T2. It can be seen that reconstructing the ventricles usually poses problems and yields a high amount of false positives.

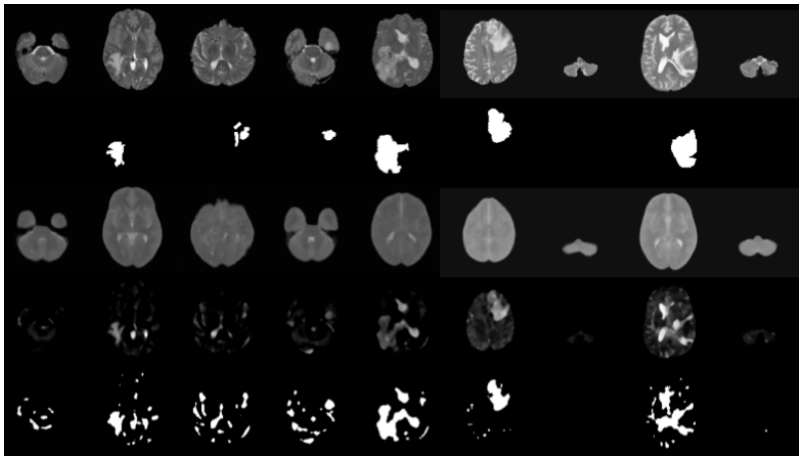


Figure 5.12: Similar to Fig. 5.11, however now on the BraTS T2 dataset. Common failure modes include the inability to reconstruct the ventricles faithfully when being trained on CamCAN T2 and tested on BraTS T2. Therefore these regions usually yield a high amount of false positives.

Fig. 5.13 shows on common failure mode that lesions which are not hyper-intense compared to the surrounding tissue can not be detected by thresholding the residual map. More powerful models combined with restoration, such as used in [Chen et al., 2020] might be able to overcome those issues.

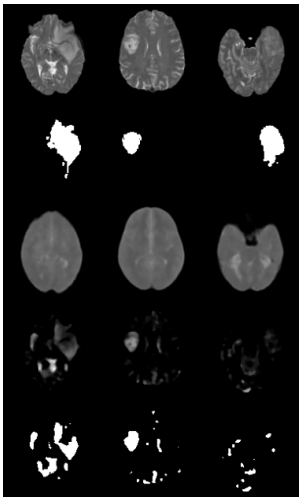


Figure 5.13: Another failure mode of the VAE based models presented in this work is their inability to detect lesions which are not hyper-intense. While the lesions associated with the first two colums (brain compared to remaining brain tissue) can be detected, the lesion in the rightmost column is not detected at all.

Histogram Matching Making sure that the patient-wise intensity histograms are matched to the patient against which the training samples have been matched against improves segmentation performance slightly (9% increase in Dice score), as can be deduced from Tab. 5.1. More importantly, it remains to examine how histogram matching effects domain-shifted data in the upcoming experiments.

5.7 Domain Shifts vs. Lesions

The following experiments are designed to shed light on the entanglement of effects on commonly used slice-wise lesion detection originating from domain-shifts, that is, training on one distribution and testing on another one, and actual lesions within an image.

From fig. 5.14 we can conclude that neither the reconstruction error nor the ELBO are able to distinguish between a lesional sample that originates from the same distribution as the training data, that is, a sample from *CamCAN T2 lesional* and a healthy sample which has undergone a domain-shift, for example originating from *BraTS T2 healthy*. This is due to the fact that unsupervised anomaly detection, exploiting reconstruction errors, is by nature principled on OOD detection. This makes it blind to actual OOD samples which is being confirmed by this experiment.

Another interesting insight gained in fig. 5.14 is that there might be OOD datasets, such as plain Gaussian noise in this case, which get mapped into the latent space in such a way that the KL divergence between this representation and the prior is actually smaller compared to the in-distribution dataset. This can happen even though the model has been trained in such a way that the latent-space representation of the in-distribution data is regularized in such a way that it lies close to the prior. This latent space encoding can change depending dramatically depending on the model architecture and loss function used as shown in experiments in Sec. 5.11. Also, note that the KL divergence of FashionMNIST, which can be regarded as heavily OOD, differs significantly compared to Gaussian noise. Thus, we conclude that it can be dangerous to rely on the KL divergence as a metric to conduct slice-wise lesion detection, even though [Zimmerer et al., 2020] reported that using the KL term for exactly that did yield the best results for their setup.

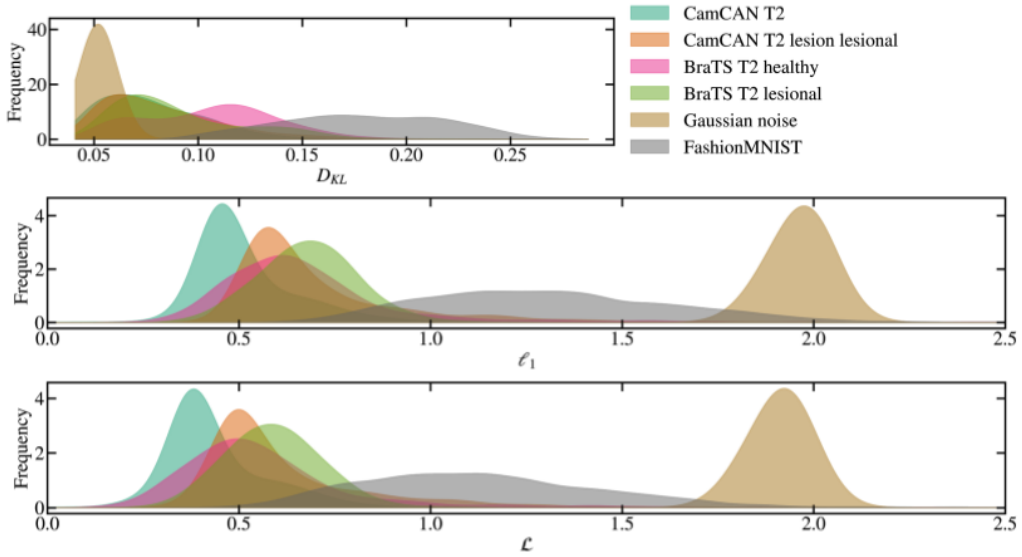


Figure 5.14: Densities for sample-wise loss term contributions (cf. 3.4) for various OOD datasets. *CamCAN T2* (teal) represents the in-distribution training data containing only healthy slices. *CamCAN T2 lesion* holds samples from *CamCAN T2* but with artificially added Gaussian blobs to simulate lesional samples as explained in Sec. 5.1 and shown in Fig. 5.11. All other datasets can be regarded as OOD. We can conclude that non of the commonly used metrics, that is, D_{KL}, l_1 (reconstruction error) or $\mathcal{L}$ is able to differentiate between whether a sample is being detected as abnormal due to a domain-shift or due to actual lesions.

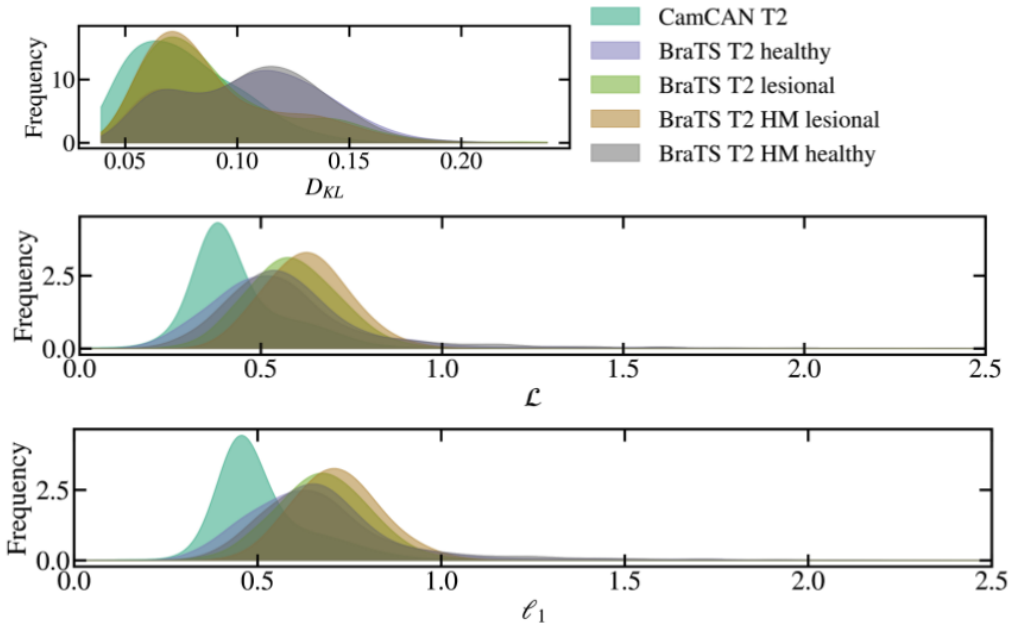


Figure 5.15: Similar setup as in Fig. 5.14, however in this case the focus lies on the effect of histogram matching and whether it can mitigate the entanglement of domain-shift effects and actual lesions. Lesional samples tend to have higher reconstruction errors compared to healthy ones on average which is expected. Surprisingly, performing histogram matching seems to increase the reconstruction error distribution slightly although a superior segmentation performance is being achieved. Histogram matching does not seem to change the behaviour in the latent space. It is noteworthy though that for this model, lesional samples have a smaller KL divergence on average compared to healthy samples.

5.8 The Entropy Score

This section shall present the various experiments conducted to assess the applicability of the normalized entropy score (cf. Equ. 4.2) as an uncertainty measure to target slice-wise lesion detection.

First, we visualize samples of the proposed entropy score in fig. 5.16 and 5.17. The former shows the standard l_1 residual while in the latter figure the l_1^{max} distance is used as introduced in Sec. 5.2.

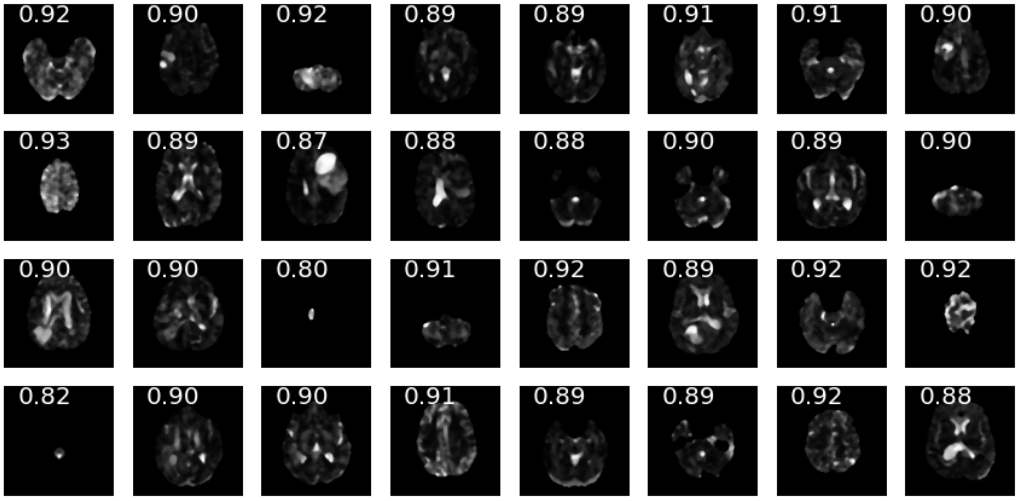


Figure 5.16: Normalized entropy scores for a number of BraTS T2 l_1 residual maps.

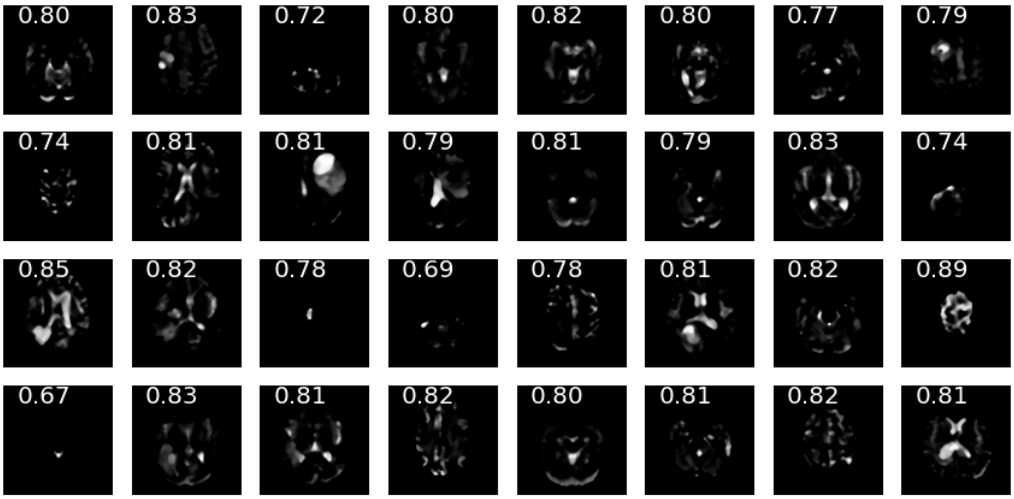


Figure 5.17: Similar to fig. 5.16, however now with the l_1^{max} score as introduced in section Sec. 5.2.

From those visual results it is indeed not possible to conclude that this is a useful metric to disentangle lesions from domain shifts. Furthermore Tab. 5.2 shows that the entropy H_{l_1} has no more discriminative power compared to the well-known reconstruction error l_1 in separating lesional slices from healthy slices.

5.9 Slice-wise anomaly detection

This section deals with the task of slice-wise anomaly detection. A system or metric which would be able to disentangle the root causes of a sample to appear OOD, that is, be it from domain-shifts or from actual lesions, is expected to perform well in detecting lesional slices. The results of employed metrics are summarized in Tab 5.2. Neither the proposed entropy score (cf. Equ. 4.2), nor the $WAIC$ (cf. Equ. 3.4) or the $DoSE$ score based on the indicated statistics (cf. Equ. 3.7) is able to outperform the reconstruction error or for that matter the KL-divergence.

Table 5.2: Slice-wise anomaly detection on various Out-of-Distribution datasets. The positive class holds lesional slices from the respective test set while the negative class holds healthy slices from the test and in-distribution validation set (CamCAN T2). To this point, neither the entropy score nor classical OOD detection metrics are able to outperform the classical reconstruction error metric.

Metric	ℓ_1		D_{KL}		$\mathcal{L}$		H_{ℓ_1}		WAIC		DoSE($\ell_1, D_{KL}, \mathcal{L}, H_{\ell_1}$)	
Test Set	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}
BraTS T2 HM	0.74	**0.44**	0.50	0.31	0.74	**0.44**	0.66	0.37	0.64	0.41	0.55	0.30
BraTS T2	0.72	0.41	0.47	0.29	0.73	**0.42**	0.65	0.35	0.62	0.39	0.53	0.29
BraTS T2 HM H-flip	0.73	**0.41**	0.50	0.30	0.73	**0.41**	0.66	0.36	0.62	0.38	0.54	0.29
BraTS T2 HM V-flip	0.59	0.31	0.38	0.22	0.59	0.31	0.53	0.27	0.62	**0.36**	0.46	0.25
CamCAN T2 artificial	0.83	**0.82**	0.53	0.61	0.82	**0.82**	0.71	0.75	0.77	0.78	0.67	0.66

5.10 Out-of-Distribution detection

Table 5.3: OOD detection performance. ℓ_1 is the mean reconstruction error per pixel for a slice, D_{KL} the KL divergence between the prior and approximate posterior and $\mathcal{L}$ the ELBO. $WAIC$ has been defined in Equ. 3.6 and $DoSE$ in Equ. 3.7

OOD Metric →		ℓ_1		D_{KL}		$\mathcal{L}$		$WAIC$		$DoSE_{(l_1,D_{KL},\mathcal{L})}$	
OOD Dataset		AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}	AU_{ROC}	AU_{PRC}
BraTS	all	0.89	0.86	0.68	0.61	0.85	0.81	0.93	0.95	0.78	0.70
T2	healthy	0.85	0.84	0.73	0.71	0.79	0.79	0.87	0.91	0.78	0.76
HM	lesion	0.92	0.89	0.63	0.55	0.90	0.87	0.99	0.99	0.78	0.70
BraTS	all	0.87	0.85	0.69	0.62	0.83	0.81	0.91	0.92	0.79	0.72
T2	healthy	0.84	0.83	0.74	0.72	0.77	0.78	0.84	0.85	0.80	0.77
	lesional	0.92	0.90	0.64	0.56	0.89	0.86	0.98	0.98	0.78	0.70
BraTS	all	0.92	0.91	0.73	0.64	0.88	0.88	0.92	0.95	0.82	0.76
T1	healthy	0.88	0.88	0.76	0.71	0.83	0.85	0.86	0.91	0.82	0.79
HM	lesional	0.96	0.95	0.70	0.60	0.95	0.93	0.99	0.99	0.83	0.76
BraTS	all	0.95	0.94	0.77	0.74	0.91	0.90	0.97	0.98	0.86	0.82
T1	healthy	0.94	0.94	0.82	0.82	0.88	0.88	0.94	0.96	0.88	0.86
	lesional	0.97	0.96	0.73	0.66	0.96	0.96	1.00	1.00	0.84	0.78
BraTS	all	0.86	0.83	0.66	0.60	0.82	0.79	0.89	0.92	0.77	0.69
T2	healthy	0.81	0.80	0.69	0.68	0.76	0.76	0.83	0.88	0.76	0.73
HM H-flip	lesional	0.91	0.87	0.63	0.50	0.89	0.85	0.98	0.98	0.78	0.69
BraTS	all	0.94	0.92	0.63	0.59	0.92	0.90	0.99	0.99	0.79	0.73
T2	healthy	0.94	0.93	0.75	0.72	0.89	0.89	0.99	0.99	0.85	0.82
HM V-flip	lesional	0.94	0.93	0.49	0.46	0.94	0.93	0.99	0.99	0.73	0.64
CamCAN	all	0.69	0.69	0.55	0.58	0.66	0.68	0.74	0.79	0.63	0.61
T2	healthy	0.51	0.51	0.51	0.50	0.52	0.52	0.68	0.73	0.52	0.50
artificial	lesional	0.87	0.85	0.60	0.64	0.84	0.82	0.81	0.84	0.74	0.70
MNIST		1.00	1.00	0.00	0.31	1.00	1.00	1.00	1.00	1.00	1.00
Gaussian Noise		1.00	1.00	0.00	0.30	1.00	1.00	1.00	1.00	1.00	1.00

Simply detecting whether a sample is OOD or in-distribution is arguably a simpler task than detecting whether the same sample is lesional or not. When comparing Tab. 5.3 with Tab. 5.2 this becomes evident and is another manifestation of the fact that domain-shifted effects are entangled

with lesional samples. Tab. 5.3 subdivides the predictions whether a sample is OOD or not into subgroups of the dataset, that is considering only lesional, only healthy or all samples. In general we can see that it is easier to detect lesional samples as OOD which is expected since the shift from the lesion is added on top a domain-shift from *CamCAN T2* to *BraTS T2*. *CamCAN T2 artificial* with artificially pasted lesions is a special case where the healthy samples have not undergone a domain-shift and in consequence are in-distribution. All classifiers but the WAIC score are complete random which is expected. The threshold for the number of lesional pixels within the brain mask is set to 20 which means that there could be a slight amount of very mildly lesional samples in the healthy set which the WAIC score could detect. Otherwise it's hard to explain why it would not report random classification. Although it seems the WAIC score is the superior metric to detect ood samples (be it due to lesions or domain-shifts), it is not superior to the reconstruction error in actually detecting lesional slices. This is reflected in the row *lesional* of *CamCAN T2 artificial* which is essentially slice-wise lesion detection. The same behaviour was observed in Tab. 5.2 where the reconstruction error and ELBO term are amongst the best performing metrics. We have to conclude that neither the WAIC score nor the DoSE score based on the proposed training statistics is effective in disentangling the influencing factors for an image to be OOD.

5.10.1 Exploring the Gaussian Annulus Theorem in the context of VAEs

Fig. 5.18 shows reconstructions which arose from random samples taken in the $d = 128$-dimensional latent space. Since the latent prior reads $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, 1)$ and the approximate posterior $q(\mathbf{z}|\mathbf{x})$ is regularized to be close to $p(\mathbf{z})$ in terms of KL divergence, the most typical samples in the latent space encoding representing the learned distribution should reside at $\sqrt{d} = 11.3$ according to the Gaussian annulus theorem (cf. Fig. 3.4). These samples lie in the range corresponding to the spheres with minimum and maximum distances to the origin of $[10, 12]$. The observations made in Fig. 5.18 do support this theory since this is the range where we observe the most typical examples. A similar effect can be observed when taking random Gaussian samples in the latent space with, essentially sampling from $\mathcal{N}(\mathbf{0}, 1)$ directly, as shown in Fig. 5.19 which is the same as sampling from the range discussed above. The distribution of the distances to the origin for 1000 random samples from a normal 128-dimensional Gaussian is shown in fig. 5.20.

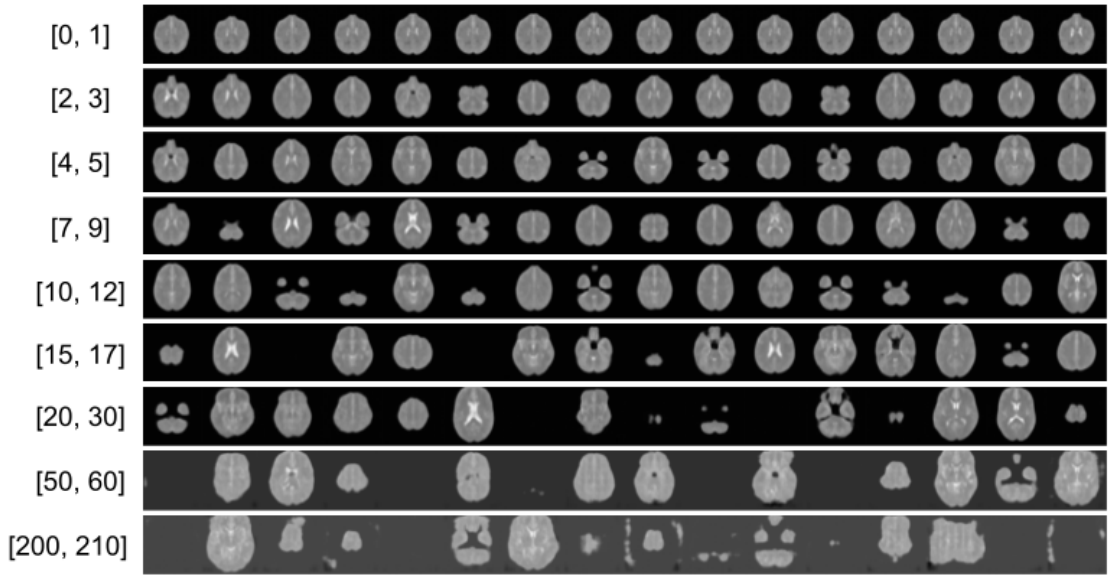


Figure 5.18: Each line holds a number of reconstructions which reside from random samples from the latent space. The numbers on the left side represent the minimum and maximum radii of a hyper-shell lying in the 128-dimensional latent space from which the corresponding samples from the right side. Since $\sqrt{d = 128} = 11.3$, this is the distance (line $[10, 12]$) from the origin where the typical samples from our dataset reside. Note that this model has been trained with $\beta = 1.0$

.

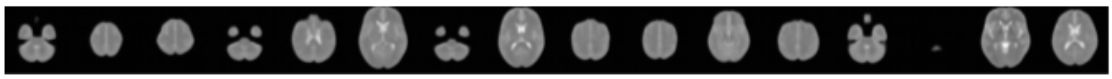


Figure 5.19: Reconstructed random Gaussian latent space samples follow the training distribution in a visually coherent manner.

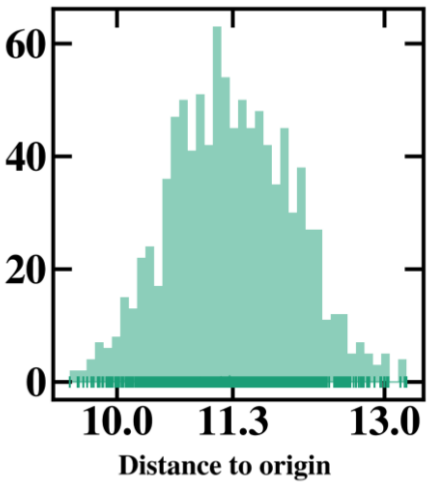


Figure 5.20: Distribution of the distances to the origin (norms) for 1000 random samples from a normal Gaussian.

.

5.11 Masked Training

As introduced in Sec. 3.1.2, the cost function of a VAE is comprised of two terms, one of which enforces compliance of the latent code with a multivariate Gaussian prior while the other one makes sure the model produces faithful reconstructions. The following experiments were conducted from a situation during the project were a flawed loss function in the implementation led to a dominating KL-term in the VAE loss function. This most probably led to a so-called posterior collapse as discussed before.

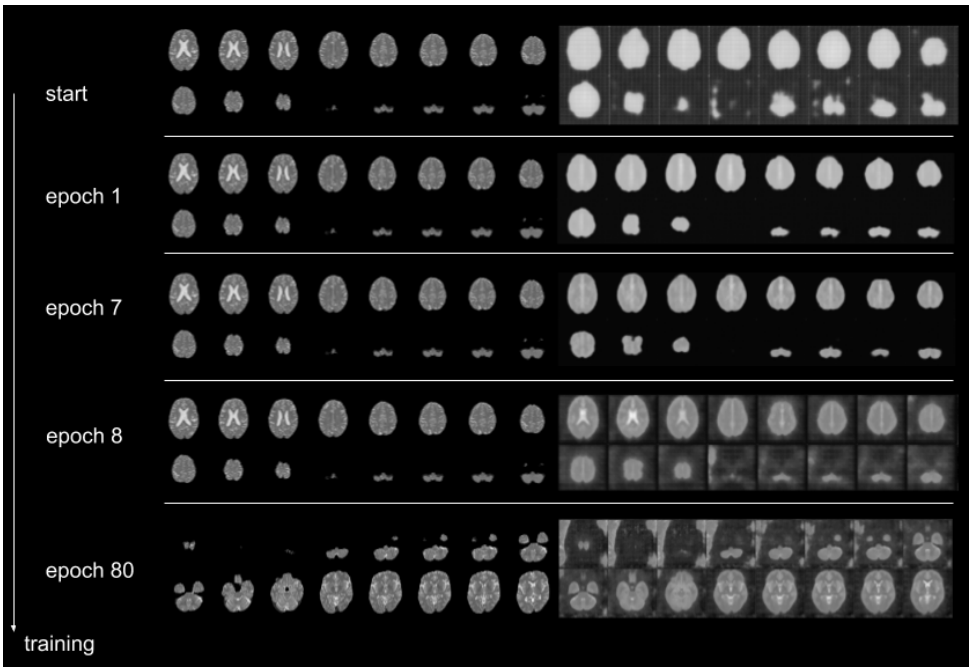


Figure 5.21: The effect of switching on masked loss calculation during training. For each epoch considered, there is a batch of 16 slices on the left and the corresponding reconstructions on the right. While in early epochs, the model learns to predict the background pixel values rather quickly (after 1 epoch), it produces lots of artefacts in later epochs after masking has been enabled (after epoch 8). Since there is a brain mask associated with every slice, evaluations can be performed on pixels within the mask exclusively. Therefore, the artefacts should not bother us too much.

Due to the geometric shape of a brain, the images might be dominated by background pixels, especially for slices at the very beginning or end in the axial direction. This might hinder training and put too much emphasis on pixels outside of the brain which are not of concern when constructing an unseen sample and checking the residual map within the brain mask. Thus, we examined the effect of masked training which means that during training only pixels within the brain mask contribute to the reconstruction loss.

We found (cf. Fig. 5.21) that, although the model produces lots of artefacts around the brain region, the reconstruction quality does, if at all, improve slightly compared to non-masked training, that is, when all pixels of an image contribute to the reconstruction loss. Note that, even though the model has learned to predict the background pixels in earlier epochs, it does produce artefacts in the background after masked training has been enabled. This is due to the fact that weights within the network are getting updated over the course of training and might affect the background pixels in which region the model is not being penalized anymore. Thus is does not matter whether masked training is switched on from the beginning or at a later point in time during the training process. There is no discrepancy in terms of segmentation performance, both variants result in the

same Dice score.

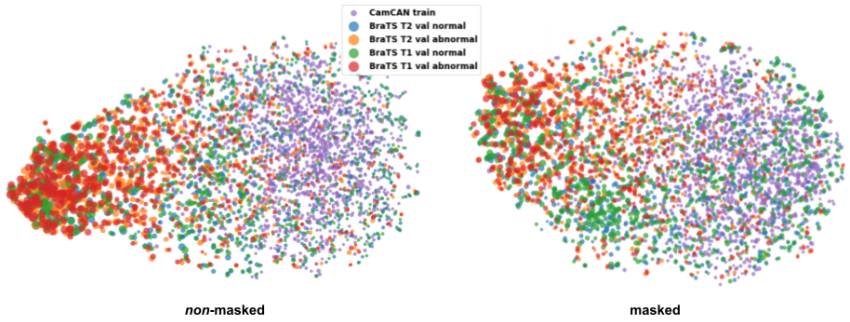


Figure 5.22: 2D UMAP embedding of latent codes from samples originating from different datasets showing the effect of enabling masked training on the latent space distribution of input samples from different datasets. The models have been trained on CamCAN T2 as usual.

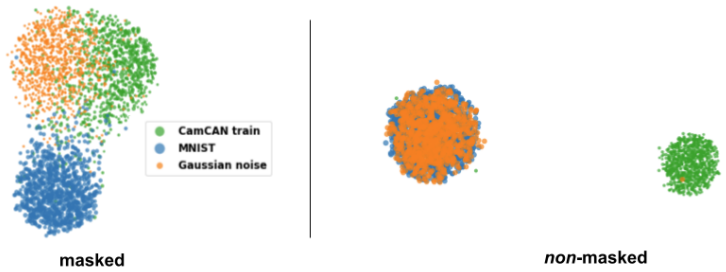


Figure 5.23: Similar to Fig. 5.22, however this time with MNIST and Gaussian noise samples only along the in-distribution CamCAN T2 dataset. The latent space embedding changes drastically which is also reflected in Fig.5.24

.

Furthermore, we observe that the latent space embedding does change significantly when masking is switched on or off (see Figures 5.22 & 5.23. While it is difficult to explain those findings conclusively, they might serve as a starting point for experiments to regularize the latent space during training in such a way that the KL-divergence can actually serve as a robust metric. With the current findings we have to conclude that it should not be used as an anomaly detection metric as done by [Zimmerer et al., 2020] due to predisposition to changing model parameters.

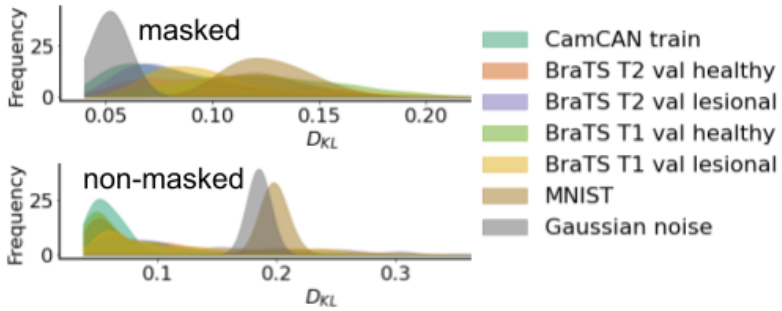


Figure 5.24: Sample-wise KL divergence histograms for masked and non-masked training.

Chapter 6

Conclusion

With this chapter I would like to conclude this thesis and recapitulate on the findings and approaches taken. It will put the results into a larger context and interpret their meaning for the safe use of unsupervised lesion techniques while at the same time propose directions for follow-up efforts.

The results in this work showed clearly that it is crucial to understand that frameworks relying on unsupervised lesion detection are vulnerable to OOD data, that is domain-shifts. In this work, we have attempted to point out and tackle this fundamental flaw in these kind of approaches. Disentangling the root causes for a sample to be marked OOD in a completely unsupervised settings is a tremendously challenging problem and it has to be questioned whether it is in fact possible to succeed in this assignment without the use of dome extent of supervision, target domain data at hand or strong opinions on the type of lesions on wants to detect.

That being said, the approaches presented in this work, namely the use of weak supervision in the form of an uncertainty-related entropy score, as well as the use of recently developed OOD detection metrics (WAIC, DoSE) did not help to improve slice-wise lesion detection or disentangle the lesional samples and domain-shift effects. The best performing metric to do so is still represented in the form of the reconstruction error or the ELBO which have been shown to suffer from entanglement with domain-shift effects.

Future work might explore approaches which incorporate OOD data directly during training to learn notions related to domain-shifts. This would deviate away from the completely unsupervised assumption limiting the deployed model to a certain domain for which training data could be supplied. Another line of work is concerned with what is referred to domain-adaptation which aims to align samples from a source domain with a target domain, in this case the domain of in-distribution training data. Those works are worthwhile exploring, especially for the case when no fully unsupervised approach can be found to mitigate the problem of entanglement. While this would again limit the range of applications, it might be a relatively cheap way to render a model usable for a particular source domain of test images.

One limitation of this work is that there is no public access to datasets which hold scans from healthy as well as lesional patients that can be regarded from the same domain apart from lesions. We overcame this issue partially by artificially crafting a dataset *CamCAN T2 lesion* which includes artificial lesions in the form of Gaussian blobs to compare anomaly detection performances to domain-shifted lesional samples. It would be desirable to see such experiments with real data in the future which potentially might only be possible in collaboration with hospitals or other data providers.

Furthermore, training generative models can be rather involved and involves manual inspection during training on the training dynamics and latent space encoding behaviour. This led to the conclusion that using for example the KL divergence term of the VAE loss function as a standalone anomaly detection metric should be done with care. It has been shown that the encoding behaviour strongly depends on the model chosen as well as on the datasets at hand. While some OOD datasets showed very large shifts away from the prior distribution, others were encoded in a way to follow the prior distribution rather well. Exploring this line of work to make the latent space encoding a more robust metric for anomaly detection would be highly desirable. This would most likely involve a regularization scheme during training to encourage a certain, potentially disentangled, representation in the latent space.

Finally, I would like to reflect on the initial objective of this work, which was the question whether it is possible to detect the root causes of a sample to be predicted OOD in the setting of fully unsupervised lesion detection. While the answer to this question is still subject to further research, I hope that the insights gained in this work, not only related to this specific task but also to model behaviour and latent space encoding, will serve as a starting point for future works to come. May this thesis be a building block of the mosaic which will eventually enable safe and trustworthy helpers in the form of deep learning models for doctors and patients around the world.

Appendix A

Additional Materials

A.1 Additional Visual Examples

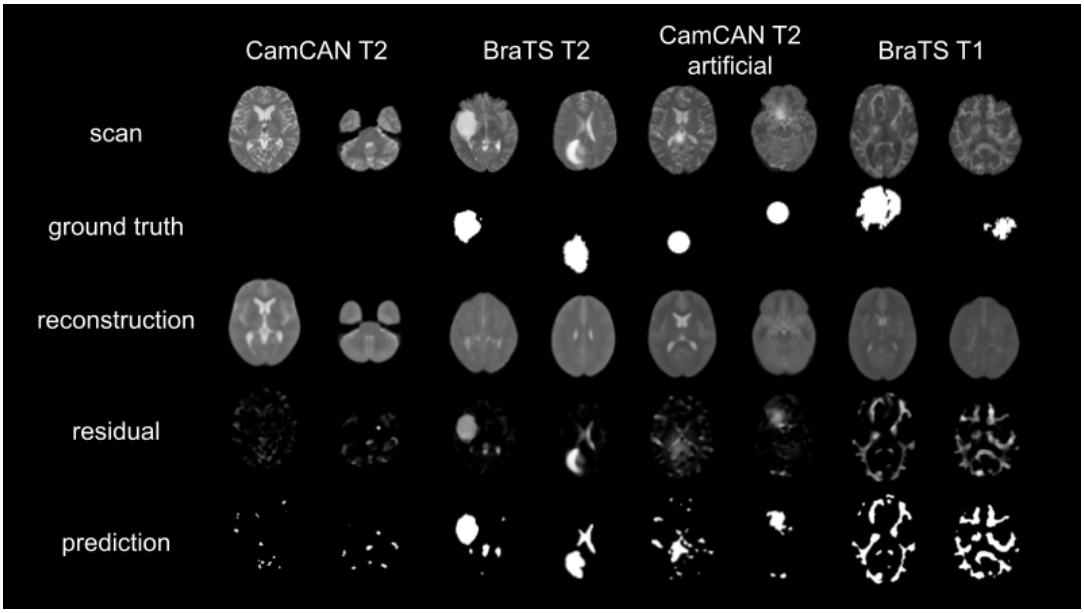


Figure A.1: Running inference using the in-distribution training data from *CamCAN T2* and various OOD datasets.

A.2 Additional Insights From Learning Disentangled Representations

The following reveals more insights from using the β-VAE framework on our MRI datasets. Fig. A.2 shows how an increased $\beta = 100$ value pushes the variation in the dataset into two distinct dimensions, indexed by 80 and 108 in this case. For this experiment, each single dimension in the latent space representation has been adjusted to run from -3σ to $+3\sigma$ while all other values are kept fix at 0. Those latent space representations are then fed through the trained encoder network to receive the reconstructions shown in the following figures.

Fig. A.3 provides a zoom into the axes which hold the variation along with two real examples and their latent space representation for those two dimensions. That way one can observe to which place in the variational spectrum they get mapped to.

Figure A.2: Slides across $\pm 3\sigma$ (vertical axis) along one non-constant dimension i (horizontal axis) of a 128-dimensional latent space. All values $z_{j \neq i}$ are hold constant at 0 for $\beta = 100$. Note that all the variation is pushed into two axes, z_{80} and z_{108} in that case.

.

Fig. A.4 shows a grid view of varied values in the two dimensions discussed above. This reveals that most, if not all, of the visual variations in the dataset can be achieved by choosing appropriate values in those two dimensions while holding all other values at 0 and feed those curated latent space representations through the trained decoder network.

While this direction of work poses an interesting path for future research with promising results (cf. Fig. A.2), we did not continue down the disentanglement learning road. The main reason being, that we reside in a fully unsupervised setting and have no access to lesional samples during training. Nonetheless, experiments in this direction, e.g. by polluting the training data with lesional samples might be worthwhile pursuing.

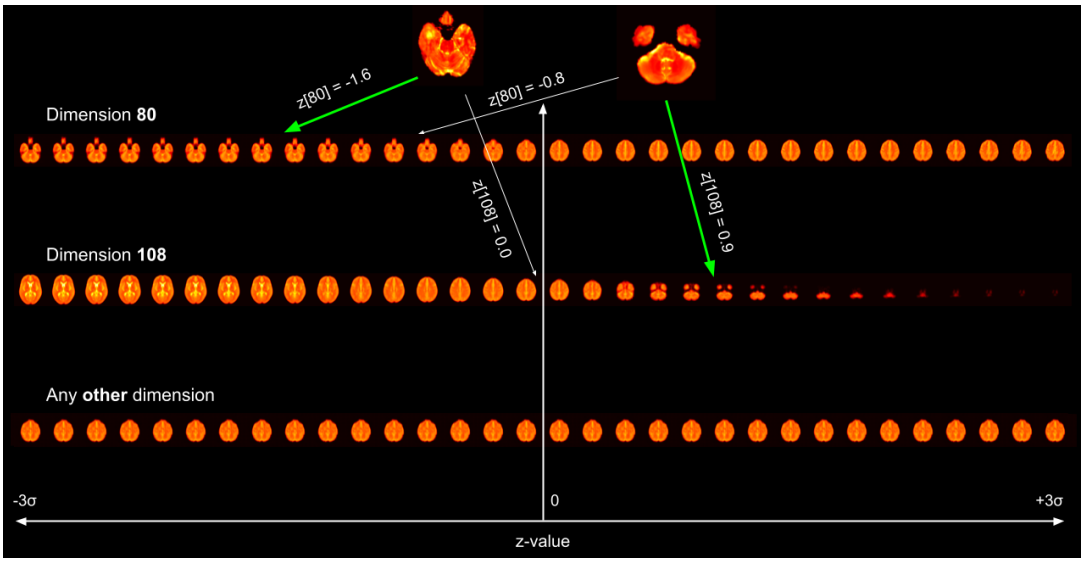


Figure A.3: Slides across $\pm 3\sigma$ (vertical axis) on the dimensions holding the variations. The two examples at the top show reconstructions from sampling at a particular latent space position in the respective dimension (green arrow). They align correctly with the passes along the dimensions holding the semantic variance

.

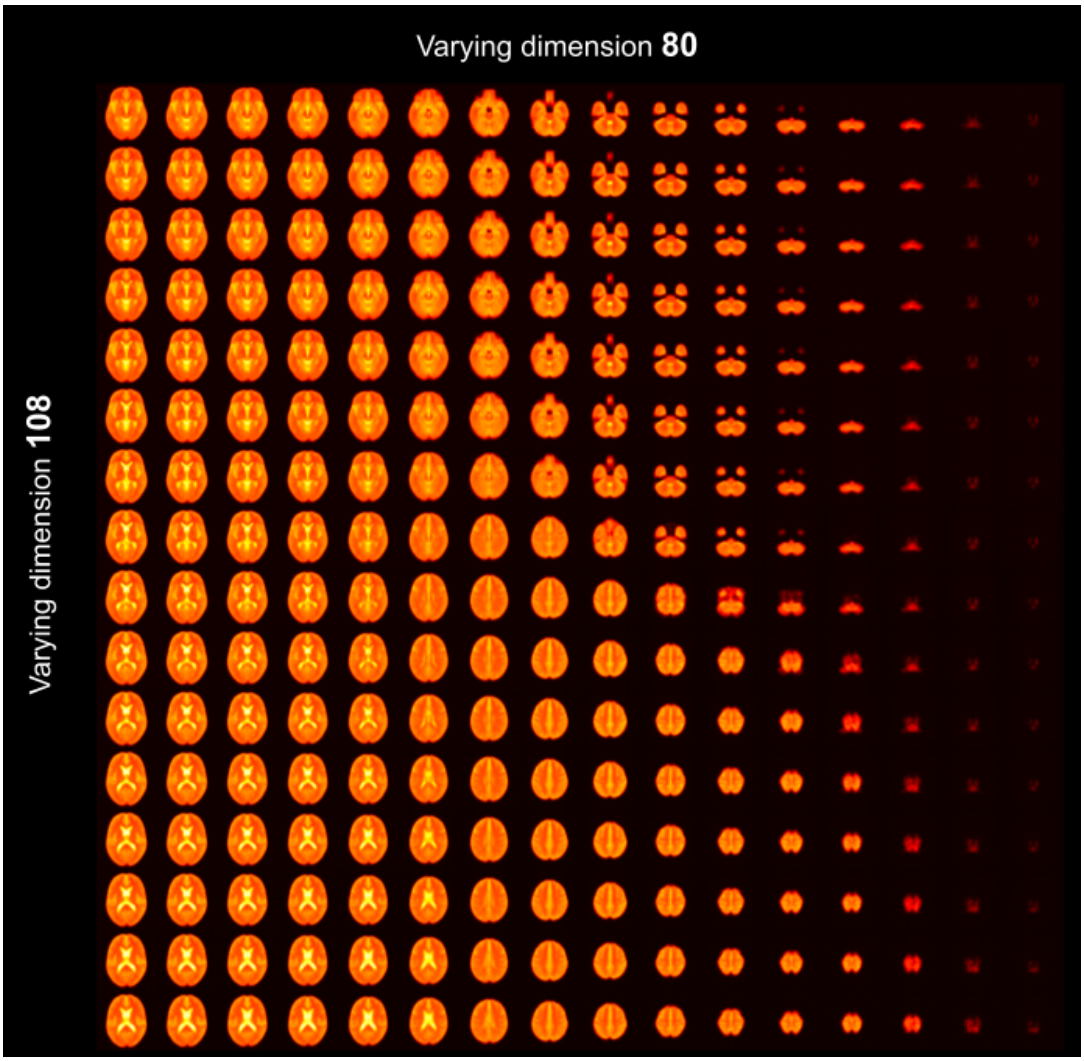


Figure A.4: Grid of two slides across two dimensions in latent space and corresponding reconstructions

.

A.3 The effect of β-Annealing

β-annealing [Bowman et al., 2016] refers to the strategy of varying the KL-term weighting (β) in the VAE cost function (cf. . 3.4) over time, that is, over the course of training. β-annealing has been used through various works deploying Variational Autoencoders. It has been shown that β-annealing helps to counteract the issue of posterior collapse. Furthermore, it has been shown that, by deploying β-annealing, more meaningful latent space representations could be created. We implemented various versions of β-annealing, shown in Fig. A.5.

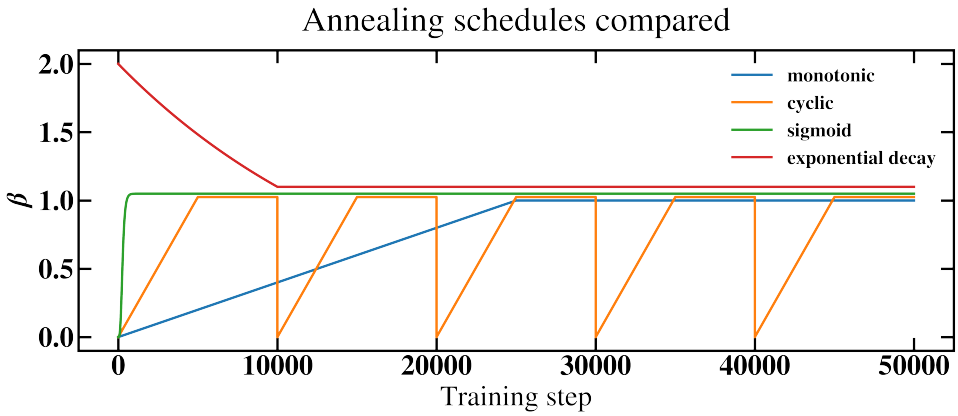


Figure A.5: Different implementations of β-annealing strategies found throughout the literate. The KL term weight, β, varies over time by (a) monotonically increasing from a start to a target value, (b) periodically increasing to a target value before dropping down again (cyclic), (c) increasing to a target value in a generalized sigmoidal shape and (d) using an exponential decay until a target value is reached.

.

The parameters in the β-annealing schedule become hyperparameters of the training procedure and have to be set by the operator. Unfortunately, using these advanced annealing schedules during training did not help to improve lesion segmentation performance. However, a low value of β in the early training stages did help to learn faithful reconstructions more quickly which is why we opted for a monotonically increasing β-annealing schedule for most experiments.

Bibliography

[Van, 2001] (2001). Automated segmentation of multiple sclerosis lesions by model outlier detection. *IEEE Transactions on Medical Imaging*, 20(8):677–688.

[Sch, 2019] (2019). f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 54:30–44.

[Aigner and Stephens, 2016] Aigner, K. and Stephens, F. (2016). *Onkologie Basiswissen*. Springer Berlin Heidelberg.

[Arevalo et al., 2017] Arevalo, O., Valenzuela, R., Esquenazi, Y., Rao, M., Tran, B., Zhu, J., Bhattacharjee, M., Fonseca, P., Doyle, N., and Riascos, R. (2017). The 2016 world health organization classification of tumors of the central nervous system: A practical approach for gliomas, part 1. basic tumor genetics. *Neurographics*, 7:334–343.

[Baur et al., 2020] Baur, C., Denner, S., Wiestler, B., Navab, N., and Albarqouni, S. (2020). Autoencoders for Unsupervised Anomaly Segmentation in Brain MR Images: A Comparative Study. 14(8):1–16.

[Baur et al., 2019] Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2019). Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. *Lecture Notes in Computer Science*, page 161–169.

[Bengio et al., 2012] Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538.

[Bishop, 1994] Bishop, C. M. (1994). Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222.

[Bowman et al., 2016] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space.

[Burgess et al., 2018] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in β-vae.

[Chen et al., 2019] Chen, R. T. Q., Li, X., Grosse, R., and Duvenaud, D. (2019). Isolating sources of disentanglement in variational autoencoders.

[Chen et al., 2016] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets.

[Chen and Konukoglu, 2018] Chen, X. and Konukoglu, E. (2018). Unsupervised Detection of Lesions in Brain MRI using constrained adversarial auto-encoders. (Midl):1–9.

[Chen et al., 2020] Chen, X., You, S., Tezcan, K. C., and Konukoglu, E. (2020). Unsupervised lesion detection via image restoration with a normative prior. *Medical Image Analysis*, 64:540–556.

[Choi et al., 2019] Choi, H., Jang, E., and Alemi, A. A. (2019). WAIC, but Why? Generative Ensembles for Robust Anomaly Detection. *ICLR*.

[Dumoulin and Visin, 2018] Dumoulin, V. and Visin, F. (2018). A guide to convolution arithmetic for deep learning.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3(January):2672–2680.

[He et al., 2019] He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. (2019). Lagging inference networks and posterior collapse in variational autoencoders.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

[Isensee et al., 2020] Isensee, F., Jaeger, P. F., Full, P. M., Vollmuth, P., and Maier-Hein, K. H. (2020). nnu-net for brain tumor segmentation.

[Kamber et al., 1995] Kamber, M., Shinghal, R., Collins, D. L., Francis, G. S., and Evans, A. C. (1995). Model-based 3-d segmentation of multiple sclerosis lesions in magnetic resonance brain images. *IEEE Transactions on Medical Imaging*, 14(3):442–453.

[Kiefer, 1953] Kiefer, J. (1953). Sequential minimax search for a maximum.

[Kim and Mnih, 2019] Kim, H. and Mnih, A. (2019). Disentangling by factorising.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Kingma and Dhariwal, 2018] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, (Ml):1–14.

[Konukoglu and Glocker, 2018] Konukoglu, E. and Glocker, B. (2018). *NeuroImage*, 181:521–538.

[krebsliga.ch, 2020] krebsliga.ch (2020). brochure krebsliga.ch "hirntumoren und hirnmetastasen bei erwachsenen". https://www.krebsliga.ch/ueber-krebs/krebsarten/hirntumoren-und-hirnmetastasen/. Accessed: 2020-12-26.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

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

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[Mårtensson et al., 2020] Mårtensson, G., Ferreira, D., Granberg, T., Cavallin, L., Oppedal, K., Padovani, A., Rektorova, I., Bonanni, L., Pardini, M., Kramberger, M. G., Taylor, J.-P., Hort, J., Snædal, J., Kulisevsky, J., Blanc, F., Antonini, A., Mecocci, P., Vellas, B., Tsolaki, M., Kłoszewska, I., Soininen, H., Lovestone, S., Simmons, A., Aarsland, D., and Westman, E. (2020). The reliability of a deep learning model in clinical out-of-distribution MRI data: a multicohort study. *Medical Image Analysis*, page 101714.

[McCulloch and Pitts, 1988] McCulloch, W. S. and Pitts, W. (1988). *A Logical Calculus of the Ideas Immanent in Nervous Activity*, page 15–27. MIT Press, Cambridge, MA, USA.

[Menze et al., 2014] Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al. (2014). The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024.

[Moraal et al., 2010] Moraal, B., Wattjes, M. P., Geurts, J. J. G., Knol, D. L., van Schijndel, R. A., Pouwels, P. J. W., Vrenken, H., and Barkhof, F. (2010). Improved detection of active multiple sclerosis lesions: 3d subtraction imaging. *Radiology*, 255(1):154–163. PMID: 20308453.

[Morningstar et al., 2020] Morningstar, W. R., Ham, C., Gallagher, A. G., Lakshminarayanan, B., Alemi, A. A., and Dillon, J. V. (2020). Density of States Estimation for Out-of-Distribution Detection.

[Nalisnick et al., 2019a] Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. (2019a). Do deep generative models know what they don't know? *7th International Conference on Learning Representations, ICLR 2019*, pages 1–19.

[Nalisnick et al., 2019b] Nalisnick, E., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. (2019b). Detecting Out-of-Distribution Inputs to Deep Generative Models Using Typicality. pages 1–15.

[Netzer et al., 2011] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

[Nwankpa et al., 2018] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378.

[Olah et al., 2017] Olah, C., Schubert, L., and Mordvintsev, A. (2017). Feature visualization. *Distill*.

[Prastawa et al., 2004] Prastawa, M., Bullitt, E., Ho, S., and Gerig, G. (2004). A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8(3):275–283.

[Ren et al., 2019] Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., DePristo, M. A., Dillon, J. V., and Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. *arXiv*, (NeurIPS).

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. , 323(6088):533–536.

[Schlegl et al., 2017] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10265 LNCS:146–147.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[Szegedy et al., 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

[Taylor et al., 2017] Taylor, J. R., Williams, N., Cusack, R., Auer, T., Shafto, M. A., Dixon, M., Tyler, L. K., Cam-CAN, and Henson, R. N. (2017). The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) data repository: Structural and functional MRI, MEG, and cognitive data from a cross-sectional adult lifespan sample. *NeuroImage*, 144:262–269.

[Tustison et al., 2010] Tustison, N. J., Avants, B. B., Cook, P. A., Zheng, Y., Egan, A., Yushkevich, P. A., and Gee, J. C. (2010). N4itk: improved n3 bias correction. *IEEE transactions on medical imaging*, 29(6):1310–1320.

[Van Essen et al., 2013] Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., and Ugurbil, K. (2013). The wu-minn human connectome project: An overview. *NeuroImage*, 80:62–79. Mapping the Connectome.

[Wolff et al., 2010] Wolff, J. E., Driever, P. H., Erdlenbruch, B., Kortmann, R. D., Rutkowski, S., Pietsch, T., Parker, C., Metz, M. W., Gnekow, A., and Kramm, C. M. (2010). Intensive chemotherapy improves survival in pediatric high-grade glioma after gross total resection: results of the hit-gbm-c protocol. *Cancer*, 116(3):705–712.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

[Zimmerer et al., 2020] Zimmerer, D., Isensee, F., Petersen, J., Kohl, S., and Maier-Hein, K. (2020). Abstract: Unsupervised anomaly localization using variational auto-encoders. *Informatik aktuell*, page 199.