Faculty of Science and Technology
Department of Physics and Technology

Segmentation and Unsupervised Adversarial Domain Adaptation Between Medical Imaging Modalities

—

Andreas Storvik Strauman

FYS-3941 - Master's thesis in applied physics and mathematics - 30 stp - July 2019

/ Abstract

Segmenting and labelling tumors in multimodal medical imaging are often vital parts of diagnostics and can in many cases be very labor intensive for clinicians [1]. The effort in advancing time-saving methods in the medical health sector might be of great help for busy clinicians and can maybe even save lives.

Furthermore, creating methods that generically, accurately and successfully process unlabelled data would be a major breakthrough in deep learning.

This thesis aims to address both these challenges by exploring and improving current methods involving adversarial discriminative domain adaptation on multimodal imaging, and address weaknesses, not only in adversarial discriminative domain adaptation, but also in the general adversarial discriminative cases.

More specifically, this thesis

1) applies CNNs to segment soft tissue sarcoma tumors in PET, CT and MRI modalities and to the author's best knowledge achieves state-of-the-art results,
2) explores unsupervised adversarial discriminative domain adaptation on segmentations of soft tissue sarcoma tumors between permutations of PET, CT and MRI and
3) demonstrates weaknesses in state-of-the-art adversarial discriminative training, and finally
4) improves and provides groundwork for further research on said techniques.

Additionally, the thesis will also provide strong fundamental background for applying adversarial discriminative domain adaptation for use in medical modalities, including a solid introduction to deep learning in medical imaging, both from a theoretical and practical aspect.

/ Acknowledgements

First and foremost I would like to address Michael Kampffmeyer PhD and Professor Robert Jenssen. Thank you for the time invested in me and for guiding me through this thesis.

To my fiancee Lillian, to my son Leon and to my family and friends: thank you for your patience, love and support.

I also would like to address two of my fellow students, and friends, Mads and Daniel. You have my greatest appreciations for the inspiration, motivation and knowledge you have given me.

At UiT, the number of people to acknowledge is too great to list. Thank you to my professors, colleagues and employers.

Lastly, to Linda, Peter and my mom, Solveig, who helped out at home, allowing me to focus and devote a few more hours to this thesis. Thank you.

- Andreas

🎓
Wow. What a ride.

/ Table of Contents

List of Figures

List of Tables

/ Algorithms

Abbreviations

ADDA	Adversarial discriminative domain adaptation.
AUC	Area under the (receiver operating characteristics) curve.
CDS	Clinical decision support.
CNN	Convolutional neural network.
CT	X-ray *c*omputed *t*omography.
DCGAN	Deep convolutional generative adversarial network.
DICOM	Digital Imaging and Communications in Medicine.
GAN	Generative adversarial network.
MCMC	Markov chain monte carlo.
MNIST	A *M*odified version of the *N*ational *I*nstitute of *S*tandards and *T*echnology (NIST) database, containing centered handwritten digits [3].
MRI	Magnetic resonance image (also only MR can be used).
PET	Positron emission tomography (also referred to as PT).
ROC	Receiver operating characteristics.
SGD	Stochastic gradient descent.
SVHN	A dataset containing *s*treet *v*iew *h*ouse *n*umbers [4].
USPS	A dataset containing handwritten text provided by the *U*nited *S*tates *P*ostal *S*ervice [5].

Part I / Introduction

1. Image analysis for medical applications

The implementation of electronic health records in United States hospitals, has risen from 8.7% in 2008 to an extraordinary 99.1% in 2017 [6, 7]. These findings strongly suggest that the amount of digitally available data within the medical health sector is growing extremely quickly due to modernization of equipment and the convenience of having digital access.

It is highly infeasible, if not impossible, to manually analyze this vast accumulation of information, and efforts are unceasingly being made to automate models to discover patterns in the data. Furthermore The American Association of Medical Colleges expects a shortage of over 130 thousand physicians by 2025 [8], and estimates that the number of doctor office visits will increase from 462 million in 2008 to 565 million in 2025 [9]. In turn, it might be of great advantage and importance to research and obtain potential time-saving methods as tools for increasing efficiency. Automated systems that provide clinicians with intelligently filtered information could be good candidates of such. Systems of this nature are often referred to as computer-based clinical decision support (CDS) systems [10].

Automating medical image analysis by combining computer science and mathematics can be dated back to at least the 1970s. Through the couple of decades that followed, medical image analysis was mostly done by hand crafted sequential algorithms, with analytic mathematics and if-else statements. Automated medical image analysis at that time was often done with sequential application of low-level pixel processing (edge and line detector filters, region growing) and mathematical modeling (fitting lines, circles and ellipses) to construct compound rule-based systems that solved particular tasks. At this early stage its main applications were detecting edges and lines, and fitting elementary shapes like circles and ellipses to perform classification and detection [11].

Machine learning has been increasingly adopted in the design of such automated systems [1, 12]. Presently, as a constituent field, deep learning has made advances leading to the development of algorithms that surpasses even expert medical professionals on multiple tasks, such as detecting skin cancer [13] and pneumonia detection [14]. Deep learning also has the advantage to discover features, relations and correlations that might not be immediately visible to humans [15]. MIT Technology Review considered deep learning one of the top 10 breakthrough technologies in 2013 [16], and as of 2016 deep learning was considered the leading machine learning tool in the general imaging and computer vision domains [17]. Deep learning algorithms have recently become a very popular approach for medical image analysis [11].

Even though adaptation of CDS systems for practical use in hospitals have been fairly slow [18], research that can prove useful in CDS systems from deep learning is increasingly being proposed in multiple fields.

As for instance in one of the author's prior publications *Classification of Post-operative Surgical Site Infections from Blood Measurements with Missing Data Using Recurrent Neural Networks* [19], where a new recurrent neural network architecture was applied to predict sepsis in patients after operations.

Object detection in images is often a vital part of diagnosing. Automatization of this process has been attempted for many years.

Within the field of deep learning, convolutional neural networks (CNNs) have particularly dominated the computer vision tasks. CNN architectures have achieved state-of-the-art status in many tasks related to computer based medical image analysis [17, 11].

The seemingly first [11] convolutional neural network (CNN) applied to medical imaging dates all back to Lo et al. in 1995 [20]. They use a mere four layer CNN for nodule[a]detection in x-ray images.

In the scientific community, CNNs weren't considered particularly useful until Krizhevsky et al. in 2012 proposed AlexNet, a Convolutional Neural Network (CNN). The purpose of this network was solving the ImageNet challenge, and

[a]Nodule: a small swelling or aggregation of cells. [21]

it did performed extremely well [22]. Following this discovery, new deep CNN architectures have frequently been proposed and deep learning as a field has dramatically improved state-of-the-art in several domains [23] and has now in some tasks even surpassed human performance [15, 24].

Today, well trained CNNs can compare and extract relevant features in a unique, flexible, robust and effective way compared compared to classical algorithms using hand-crafted features. Specifically, CNNs have produced exquisite performance in object detection, segmentation and classification of images. All of these tasks can often be considered vital parts of diagnosing and can in many cases be very labor intensive for clinicians. Additionally, said tasks are often too complex to represent and solve by using analytical equations as done by the primitive aforementioned algorithms [1]. Modern CNNs are particularly proficient when presented with 'normal' 2D pictures image analysis tasks. Said tasks usually consists of variations within classification, segmentation and detection. These properties makes them exceptionally interesting for use in medical images for CDS systems.

All of these tasks have multiple use cases. For instance classification of the image as a whole could be useful for e.g. classifying malignant or non-malignant tumors. When doing detection the objective is to localize structures in the image space, for example detecting tumors in 3D PET scans. This differs from pure classification since it is looking for whether (and often where) an object is located in an image. In segmentation, the purpose is to provide a region-wise classification, often pixel by pixel, whilst retaining a semantic structure.

A diverse selection of image generation methods have been proposed using deep learning architectures [11]. These generative models can have many useful applications in medical imaging. A few examples would be removing noise or obstructing artifacts, normalizing, improving the overall resolution or colour quality and domain adaptation. One family of such generative models is called generative adversarial network (GAN), and is often implemented using convolutional neural networks.

Most of the aforementioned results are mostly on data where the algorithms have a priori knowledge about what it wants to learn (supervised learning). To get this category of data (labelled data) medical experts have to make it, as for example manually segmenting tumors on a PET image or continuously reporting how the patient is doing. All of this also has to be done in a fashion so that the algorithms can understand it.

A great deal of data, especially in the medical sector, does not have ground truth, and will therefore be incompatible with supervised learning methods. So called unsupervised methods are getting increasingly better and are in the works. An interesting approach to working with unlabelled images is through a process called domain adaptation. Domain adaptation utilizes information obtained by the available labelled data, and applies this to unlabelled data. Examples would be images from different scan qualities, different modalities, images from different hospitals. An alternative approach for dealing with unlabelled data would be turning to fully unsupervised methods, and deep clustering analysis.

Unsupervised methods, for reasons that will be introduced later, can be regarded as unreliable and unpredictable in interpretation, and thus might be too insecure for medical use. Even though a great deal of medical images are unlabelled, not much work has been done with unsupervised methods medical imaging, or in health data in general. This makes it all more important to explore the possibilities and limitation of this field.

Unsupervised deep learning is still considered a field in it's infancy. It is the perception of the author that there is a fair amount of untapped potential in this field, and also related to it's applications in medical imaging.

1.1. Generative models in medical image analysis

Generative models may appear to have a rather restricted variety of useful applications. However, generative models can be used in a much greater extent than generating entire images, or artificially sample from distributions. As will be apparent, generative models are both used independently as methods, but seemingly more frequently used as an intermediate step for many tasks in for example domain adaptation, image reconstruction, segmentation and subsample selection.

A diverse selection of image generation methods have been proposed using deep learning architectures [11]. These generative models can have many useful applications in medical imaging. A few examples would be removing noise or obstructing artifacts, normalizing, improving the overall resolution or color quality and domain adaptation. One family of such generative models is called generative adversarial network (GAN), and is often implemented using convolutional neural networks.

One of the tasks that are amongst the most unmistakeable fit for GANs is image reconstruction. Reconstructing images is very useful in medical imaging. Removing unwanted noise or correctly interpolating or estimating content of incomplete images, could prove extremely valuable assets. Yang et al. [25] did image reconstruction by removing unwanted elements (and thus generate background) by suppression of bony structures in X-ray. Oktay et al. [26] reconstructed high-resolution cardiac MRI from one or more low-resolution MRI volumes.

GANs have been used as a instrumental part of other complex models. In such cases, the GAN often has the function of learning loss functions that express measurements that are intractable to obtain manually, compared to for example higher-order statistics.

In domain adaptation, GANs can be used for training a discriminator to obtain invariant features across domains, like in Ren et al. [27]. 2D X-ray images have much clearer structures and sharp boundaries compared to CT X-ray. Zhang et al. [28] propose an unsupervised domain adaptation application with a GAN that learns, using CT scan data, to parse anatomical objects from unlabelled X-ray images.

GANs in medical imaging are of course also used for image generation. PET and X-ray scans exposes patients for big doses of radiation, which is harmful [29], whilst MRI does not at all. Thus being able to generate PET and CT (X-ray) scans from MRI would prove very preferable over having to do potentially damaging scans. Li et al. [30] showed in 2014, by generated PET images from MRI images using GANs, that estimating PET from MRI works for diagnosing Alzheimer's disease when the PET images are missing. Image-to-image translation from MRI to CT was later done by Nie et al. [31] in 2016. GANs are also used for improving e.g. semantic segmentation [32] and Dong et al. [33] brings this one step further to unsupervised domain adaptation, and applying it to estimate the cardiothoracic ratio through segmentation of chest X-ray images.

GANs continues to be a very popular topic today and is the subject in many publications accepted by high-end conferences. For example Mahapatra et al. [34] used a conditional GAN (cGAN), combined with a bayesian neural network [35] as a feature generator in an active learning select the most informative samples for training data. The cGAN was used to generate chest x-ray images with different disease characteristics, conditioned on real images.

1.2. Challenges in medical images for CNNs

Medical images posses a set of typical characteristics that presents a particular set of challenges to classical CNN architectures. Following is a general overview of the challenges that affects the CNNs known abilities and overall performance, as it is useful to have an oversight of the role and obstacles that convolutional neural networks are facing.

The requirement for both local and global context for doing satisfactory classification, segmentation and detection, might be necessary to a greater extent in medical images. For instance when classifying a tumors malignancy, the region around said tumor could simultaneously be equally important as a microscopic image of a small part of the tumor. This could possibly be due to a result of individual variations between patients or the nature of the object or lesion to be processed. This challenge has been mainly dealt with by using multi-stream networks in a multi-scale fashion [11, 36, 37]. That is feeding multiple cropped and/or zoomed sub sections of an image into multiple parallel CNNs.

An additional noteworthy challenge is revealed when considering the technical aspect. In contrast to earlier mentioned natural coloured 2D images considered in computer vision, medical images are often images depicting spatial 3D data. Consequently, correctly incorporating 3D information is often necessary for good performance within detection, classification and segmentation. The images are typically represented as sequences along one of the axes, consisting of orthogonal 2D slice snapshots at intervals along the third axis. Additionally, some medical images like hyper spectral images might have more than the default 3 channels that corresponds to color in natural images.

Even though CNNs can process 3D images with minor modifications of the architecture, those modifications alone are often not enough to sufficiently utilize the spatial 3D information.

There have also been many different approaches to effectively take advantage of the 3D information in the image. Setio et al. [36] proposes a multi-stream CNN to binary classify points of interest in chest CT. Nie et al. [38] trains a 3D

CNN to assess survival in patients suffering from high-grade glioma tumors in the brain.

Another popular approach, especially for landmark localisation and segmentation, is to treat the 3D data as sequential orthogonal 2D planes Litjens et al. [11]. Even though it does not exclusively apply to medical images, another challenge, primarily in regards to object classification, is the unbalanced datasets. During object classification, typically each individual pixel is classified and frequently only a small portion of the images contains a small segment with the object on it. This will result in a huge class imbalance with a training set which is rather heavily slanted towards the non-object class. The non-object class is often even further extended when working 3D images, since only a small portion of these images yet again contains the region of interest.

1.2.1. Segmentation

Segmentation is the most common subject amongst papers applying deep learning to medical imaging [11], and is also common in both natural image analysis as well as in medical imaging. Typically, segmentation happens pixel wise, so that every individual pixel is classified whilst ideally retaining a semantic structure.

For capturing global contexts of the image, often so called fully convolutional neural networks, (or fCNNs), are used. That is when the last (often fully connected layer) is replaced by another convolution layer with a high receptive field.

fCNNs in their original formulation causes a drastic decrease in resolution. There are several methods proposed to counteract this resolution decrease [11]. Long et al. [39] presented a method called 'shift-and-stitch', which is fCNN applied to shifted versions of the input image. Then at the end the results are stitched together to obtain a better full resolution version of the final output.

This idea is brought one step further by the one of the most well known [11] CNN architectures for segmentation of medical images: U-net by Ronneberger et al. [40]. U-net has an upsampling part where 'up' or 'transposed' convolutions are used to increase the image size. Although this is not the first paper to introduce the transposed convolutions, Ronneberger et al. also incorporated ‘skip-connections’ to connect opposing contracting and expanding convolutional layers. The other novelty in U-net was the equal amount of up sampling and down sampling layers.

Boundary detection would be a branch of segmentation, and detection of boundary points required for model based segmentation is very challenging for organs with inhomogeneous appearance on images. For example in MRI, the contrast of the image for one organ might be very variational due to the use of difference settings and scanners at different clinical institutions. Brosch et al. presented PROMISE12, where they used CNNs to obtain state-of-the-art performance in boundary detection [41].

As earlier mentioned, working with 3D medical images with CNN presents challenges per se, but segmentation entails the extra intrinsic challenge: annotation of the data. If the data is not qualitatively annotated in sufficient quantities, it will likely prove difficult to produce 3D segmentation outputs with sufficient quality to be useful. Since medical images often are represented as dispersed slices, the annotated samples would be sparse due to inadequate coherency of the annotated areas. To address this, Çiçek et al. [42] proposed a structure based on U-net that provides full 3D segmentation from sparse 3D annotated data.

Also, standard images from modalities like MRI, PET and CT is represented as 2 dimensional orthogonal dispersed slices. The literature seems to attain the best results when, in these cases, consider the images as multiple 2 dimensional images and disregard the 3 dimensional information in these modalities[11].

Ultrasound also entails challenges for CDS systems, especially for orthopaedic surgery procedures e.g. high amounts of various imaging artefacts and a low signal-to-noise ratios. Wang et al. [43] uses what they call a pre-enhancing convolutional neural net, and uses the U-net [40] for segmentation and attaches a dense layer from one of the hidden U-net layers to simultaneously do segmentation and classification. They call the U-net variation cU-net.

1.2.2. Classification

CNNs have achieved state-of-the-art performance in multiple tasks. A fairly recent example is the CNN architecture CheXNet by Rajpurkar et al. [14] that achieved state-of-the-art performance in classifying different 14 diseases from chest x-ray images and even surpasses human performance in detection of some of these.

Hosseini-Asl et al. [44] used a 3D CNN in an autoencoder architecture and achieved state-of-the-art results in early diagnosing patients with Alzheimers disease (versus no cognitive impairment) from MRI images. Another CNN architecture with state-of-the-art diagnostic abilities is proposed by Pratt et al. [45] and it diagnoses diabetic retinopathy from digital photographs of the fundus of the eye.

2. Contributions and novelties

This section of the thesis intends to give the reader an impression of what the contributions and novelties proposed in this thesis are. Here segmentation, unsupervised domain adaptation, momentum reset scheme and confusion score will be outlined individually.

The contributions in this thesis are applying adversarial discriminative domain adaptation on the permutations of the three medical image modalities: PET, CT and MR. Segmentation of these three modalities is also presented and attains state-of-the-art result on all of three tasks.

In addition to the contributions stated above, two novelties will be presented, namely momentum reset and the confusion score. They will be discussed in depth in Part III, but the fundamental ideas, motivations and rudimentary overview is presented below.

2.1. Segmentation and unsupervised domain adaptation

This thesis provides contributions on segmentation of soft tissue sarcoma tumors on PET, MR and CT modalities. To the author's best knowledge, the results of the segmentations are state-of-the across all three modalities.

Subsequently all permutations of all three domains from the same dataset will be presented through the unsupervised domain adaptation algorithm ADDA [46]. This as well, to the author's best knowledge, has not been done before.

2.2. Novelties

Unsupervised discriminative adversarial approaches are notorious for being unstable in training and have issues with convergence [47, 48, 49, 50]. Below are two proposed novelties that will address both these issues.

2.2.1. Momentum in adversarial discriminative training

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks by Radford et al. [50], with 3840 citations on google scholar [51], is a very well known article on DCGANs. They propose a set of constraints on the architectural topology of convolutional GANs which clearly improves results drastically. Radford et al. also mentions that GAN work performed prior to their article had used simple momentum to accelerate training, while Radford et al. uses the Adam [52] optimizer, however notably with lower then the proposed momentum value.

Even though the use of moment based optimizers still are necessary to obtain good results, it is the contention of this thesis that the momentum term used in the optimizers in state-of-the-art adversarial discriminative models, disrupts training due to the effect of the cross-interaction between the adversarial parameters. Specifically that the training of one adversarial might drastically change the loss landscape of the other adversarial, which consequently leaves the momentum terms pointing in the wrong direction.

2.2.2. Confusion score

In the literature, determining when the training of an unsupervised adversarial discriminative model is complete, has been at a extrema of one of the the calculated adversarial loss functions, after a set number of iterations or some early stopping convergence criteria that also often is based on the adversarial loss.

However as the loss function of one adversarial is dependent on the other, the resulting loss function is not monotonic. For instance, when the a discriminator has trained multiple steps, the loss of the generator would likely increase as the discriminator has improved its performance.

Therefore Section 12 proposes a new stopping criterion, which takes advantage of the theoretical aspects of convergence presented in Goodfellow et al. [53].

3. Hypotheses

In addition to the segmentation and unsupervised domain adaptation of medical images modalities, this thesis will attempt to empirically demonstrate two hypotheses based on the arguments that are outlined in two foregoing sections (2.2.1 and 2.2.2), and further described in Part III in sections 12 and 11:

Hypothesis *I*: The typical use of momentum in adversarial discriminative training could in many cases cause the loss landscape of one adversarial to drastically change due to the parameter updates of the other. Therefore, periodically resetting the optimizer parameters, as proposed in the momentum reset scheme described in Section 11, will allow the optimizer to regain momentum in the correct direction, and in this way tend to perform better than without resetting momentum.

Hypothesis *II*: Using the confusion score presented in Section 12 to determine the optimal stopping point for the discriminator will in many cases produce better results than the common stopping criteria that is based on adversarial loss value.

Hypothesis *I*: will be tested using the proposed novelty ‘momentum reset scheme’, which suggests to reset the accumulated optimizer parameters during training. The momentum reset scheme will be described more thoroughly in Section 11.

Hypothesis *II*: will be tested by comparing the proposed confusion score (26) with the other common measurements, namely discriminator loss and loss convergence (after many iterations). The confusion score will be motivated and explored in Section 12.

4. Thesis outline

The thesis is composed of five parts: Part I: Introduction, Part II: Theory and related work, Part III: Novelties, and finally Part IV: Experiments.

Part II will introduce the fundamental, and most important concepts within machine learning theory. The theory here is specifically focussed on methods applied in the contributions of the thesis, namely segmentation, unsupervised domain adaptation and the two novelties. This theory is composed of a thorough introduction to to the fundamentals of training in Section 6, including an overview and definition of supervised and unsupervised learning. Consecutively in Section 7 neural networks, the heart of deep learning, are introduced. Neural networks motivate one of the core architecture used in the experiments, CNNs which will be presented in Section 7.3. The theory ties together when adversarial discriminative models are described in Section 9 and Section 10. Finally ADDA [46] is presented in Section 10.1, which is one of the core models used in the main contributions in the thesis. During the presentation of ADDA, the the algorithm is applied to two 'simple' datasets, for demonstration purposes.

With the applicable theoretical background is presented, Part III will explore a more thorough motivation of the proposed novelties, namely momentum reset in Section 11 and confusion score in Section 12.

Part IV is an overview and interpretation of the results from experimental contributions. It first presents the experimental setup in Section 14, subsequently for segmentation in Section 14.2 an ADDA in Section 14.3. In Section 15 the results of segmentation and adversarial discriminative domain adaptation are stated, interpreted and discussed, also in relation to the performance of the proposed novelties.

The thesis concludes with Part V. Here aspects of the novelties are discussed and future work and ideas are proposed, followed by concluding remarks.

Part II / Theory and related work

5. Notation

As definitions might not be consistent in the literature, and the author might differ from conventions, it is important to define the terms. This way it is easier to express equations and other terminology in a clear and unambiguous fashion. This section provides a reference describing notation used throughout this thesis unless otherwise specified or clear from context.

General typesetting:

a	Any scalar.
$\mathbf{a}$	A vector.
a_i	The value of a defined vector $\mathbf{a}$ at index i.
$\mathbf{A}$	A matrix.
$A_{i,j} = \mathbf{A}[i,j]$	The value in a matrix at row i and column j.
$\boldsymbol{\mathsf{A}}$	A Tensor.
A	Domain.

Specific symbols:

$\boldsymbol{\theta}$	Denotes parameters for a function.
$\mathbf{x}_i$	Vector containing one sample from a distribution.
$\mathbf{X}$	Matrix containing samples from a distribution $\mathbb{p}$; $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots]$.
$\mathbf{y}$	Desired output / labels.
$\mathcal{A}$	Performance measure.

Operators:

$\mathbb{E}$	Expectation.
$\mathcal{J}$	Jacobian operator ($[\mathcal{J}_{\mathbf{x}} f]_{i,j} = \frac{\partial f_i}{\partial x_j}$).
$\frac{\partial \mathbf{x}}{\partial \mathbf{y}}$	Short hand for jacobian: $\frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \mathcal{J}_{\mathbf{y}} \mathbf{x}$.
$\not\nabla_{\mathbf{w}[i,j]} \mathbf{A}$	Elementwise gradient on $\mathbf{A}$ with respect to $\mathbf{w}_{i,j}$.
$\odot$	The Hadamard product (elementwise multiplication).

Other:

$\mathbb{p}, \mathbb{q}, \mathbb{b}, \mathbb{d}$	Arbitrary probability distributions.
$\mathbf{x} \in \mathbb{p}$	$\mathbf{x}$ is a sample from $\mathbb{p}$.
$\mathbf{a} \in \mathsf{A}$	$\mathbf{a}$ is a vector in the domain (or set) A.

A note on random variables

Commonly, random variables are clearly separated from realized variables. However, this thesis does not address probability theory in a way that requires random variables to be distinguished from realized variables. To enhance readability, stochastic variables are not addressed as such. In this regard, the notation $\mathbf{x} \sim \mathbb{p}$ means that, unless otherwise clear from context, the vector $\mathbf{x}$ contains from *realized* values distributed with distribution $\mathbb{p}$.

More formally: let Y_i be a stochastic variable distributed as $\mathbb{p}$, and with corresponding values y_i when realized. Furthermore let the vector $\mathbf{y}$ contain these variables when realized: $\mathbf{y} = \{y_1, \ldots, y_n\}$, then this will be denoted in the thesis as

$$\mathbf{x} \sim \mathbb{p} \text{ or } \mathbf{x} \in \mathbb{p},$$

and should, unless the context suggests otherwise, be read as 'realized samples, $\mathbf{x}$, distributed as $\mathbb{p}$' and 'realized samples $\mathbf{x}$ that originated from distribution $\mathbb{p}$' respectively (the distinction is mostly philosophical). The same notation is applies for random vectors, and random matrices as well in a similar fashion.

6. Fundamentals of training

Statistics is considered the anchor of machine learning theory, and statistics is (at least usually) expressed through mathematics. As this thesis aims to show both supervised segmentation and unsupervised domain adaptation using deep learning, it is important to first present the underlying theory of these aspects.

The most fundamental theory will be presented in these following sections. Namely how an architecture 'learns' anything, common approaches to make it perform well and show why deep learning differs from pure optimization theory in classical statistics and mathematics.

Firstly supervised learning will be address, followed by overfitting, and an introduction to gradient descent, momentum and the Adam [52] optimizer. Subsequently, unsupervised learning will be defined and the main ideas behind unsupervised learning will be outlined.

Finally, this section will introduce two commonly used loss functions, and conclude with a thorough overview of common performance measures.

6.1. Supervised learning and Classification

Consider a vector $\mathbf{x}_i$ with an accompanying paired value $\mathbf{y}_i$. Suppose that $\mathbf{x}_i$ has a probability distribution $\mathbb{p}$ and that $\mathbf{y}_i$ is distributed conditionally on $\mathbf{x}_i$: $\mathbf{y}_i|\mathbf{x}_i \sim \mathbb{q}$. The ideal goal in supervised learning is, if $\mathbf{y}_i$ and $\mathbf{x}_i$ are considered random vectors, to obtain the expectation $\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$ for any $\mathbf{x}$ in its distribution. In other words, a map f^* is sought $f^* : \mathbf{x}_i \mapsto \mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$, or notationally easier, $f^* : \mathbf{x}_i \mapsto \mathbf{y}_i$, with objective that f^* holds for all possible samples in the distribution $\mathbb{p}$.

The challenge is that f^* might not even exist, and even if it did, in the vast majority of non-trivial cases, one will not predict exactly how it behaves or have any chance of obtaining a tractable analytical expression for it.

However, if we assume that empirical samples from the distribution $\mathbb{p}$, and their paired values $\mathbf{y}_i$ are available, then this would of course mean that empirical information about f^* also is available. We utilize this empirical data to obtain a different map f, with parameters $\boldsymbol{\theta}$, where hopefully f and f^* is as similar as possible when evaluated at *any* $\mathbf{x}$ vectors contained in the *true* distribution $\mathbb{p}$. This might be formalized as

$$\min_{f,\boldsymbol{\theta}} \left| f(\mathbf{x}, \boldsymbol{\theta}) - f^*(\mathbf{x}) \right| \ \forall\, \mathbf{x} \in \mathbb{p}. \tag{1}$$

The conventional strategy in supervised learning is to choose a fairly generic function f with many associated parameters $\boldsymbol{\theta}$. Nevertheless, choosing any generic function for f and strictly minimize with respect to $\boldsymbol{\theta}$ will not solve the problem. The important thing to note is that the goal is to optimize (1) for all the data in the *distribution*. Thus directly obtaining the actual extrema of f given *some* of the data, will fail this crucial objective.

In deep learning, and in many machine learning algorithms, it is common to choose a performance measure $\mathcal{A}$, in addition to f, that is optimized *indirectly*. Indirectly here means that the model itself should not use $\mathcal{A}$ to update its parameters, but $\mathcal{A}$ will rather be used for other important decisions as model evaluation and stopping criterion. Thus the task becomes to obtain the optimal parameters $\boldsymbol{\theta}^*$ in accordance with (1) and $\mathcal{A}$. An example of such a performance measure is percentage of correctly classified data points (accuracy). More of these performance measures will be presented Section 6.6. Optimizing of the parameters $\boldsymbol{\theta}$ is commonly performed by an algorithm that belongs to a group of algorithms called gradient descent.

To directly optimize the parameters, a distance (or alternatively a similarity) measure is usually defined. This measure will be referred to as the 'error function', $\mathcal{E}$. The error function is typically a function that maps the input data and the labels to a number:

$$\mathcal{E} : (\mathbf{x}_i, \mathbf{y}_i) \mapsto c,\ c \in \mathbb{R}$$

$\mathcal{E}$ will be the function to compare the empirical evaluations of f and, the ideal function, f^*. From here on out, the empirical evaluation of f^* might be denoted mathematically as $\mathbf{y}_i = f^*(\mathbf{x}_i)$. And similarly for the estimation $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$. $\mathbf{y}_i$ is referred to as the 'end goal', output value, 'label' or 'ground truth', and $\hat{\mathbf{y}}_i$ are referred to as ‘predictions’ or ‘model outputs’. This error function, evaluated at $\mathbf{y}_i, \hat{\mathbf{y}}_i$, will be denoted here as $\mathcal{E}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \mathcal{E}(\mathbf{y}_i, f(\mathbf{x}_i, \boldsymbol{\theta}))$ or simply as $\mathcal{E}$.

For de facto purposes, the optimization problem is often formulated as a minimizing problem of this intermediate function [54]:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{E}(\mathbf{y}_i, f(\mathbf{x}_i, \boldsymbol{\theta})),$$

where $\mathcal{E}$ is still interpreted as a dissimilarity or distance measure. As we wish to minimize $\mathcal{E}$ it is reasonable, and common, to favor functions with property of being strictly monotic and where a global minima exists.

As to not restrain the optimization process to deal with one datapoint at a time, we can define the loss function as an expectation over the distributed datapoints. The (ideal) loss function J^* is defined as:

$$J^*(\boldsymbol{\theta}) \equiv \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[\mathcal{E}], \tag{2}$$

and accordingly, the goal could be written in terms of this loss function:

$$\underset{\boldsymbol{\theta}}{\text{optimize}}\, J^*(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[\mathcal{E}]. \tag{3}$$

However, the ideal loss function, J^*, is not directly attainable[b]. Hence the loss function is often estimated (by the law of large numbers [55]) with the average of all the errors:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{n}\sum_i \mathcal{E}_i(\mathbf{y}_i, \hat{\mathbf{y}}_i). \tag{4}$$

The loss function in (4) is what will be considered when the term ‘loss function’ is used throughout the thesis.

Commonly in the literature, loss functions are non-convex functions, and thus are to be optimized with minimizing [56, 57]. This leaves us with the following goal: to obtain a function f with parameters $\boldsymbol{\theta}$ that represents $\mathbb{E}[\mathbf{y}_i|\mathbf{x}_i]$ as identically as possible. In other words: find a machine learning architecture and obtain the parameters $\boldsymbol{\theta}^*$.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J.$$

[b]Unless the true distribution of the data is known, in which case applying machine learning would be redundant

6.1.1. Overfitting

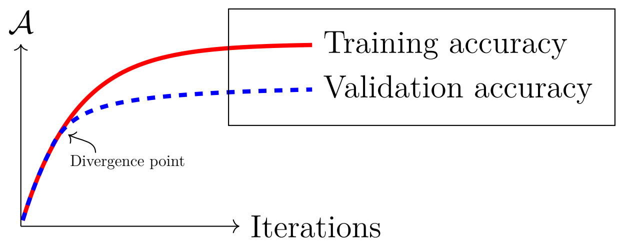


Figure 6-1: *Conceptual sketch of accuracy differences due to overfitting. When passed the deviation point, the algorithm is 'learning the data set' and not the distribution.*

If one were to disregard the performance measure $\mathcal{A}$ when obtaining $\boldsymbol{\theta}^*$ and only acquire the appropriate extremas of $\mathcal{E}(\mathbf{y}, f(\mathbf{x}_i; \boldsymbol{\theta}))$ by applying only the empirical data, the resulting map would often provide close to perfect performance scores on said empirical data samples, but would have significantly poorer scores on unseen data. This effect is conventionally referred to as overfitting.

There are actions that can be taken to help prevent overfitting, and some are discussed in Section 8 of this thesis. However, the most impactful step is likely performing a subsampling of the data during training. Due to the optimisation of the model being done iteratively, a very common approach is to split up the available data into several sets (training, validation and test). Training and validation are used during training, and test is data that is reserved for doing a final evaluation of the model.

First the gradient descent algorithm is applied to the model only evaluating the training data. When training has concluded, the performance score between train and test data will be compared. One would expect the training and test performance to be similar as well as hopefully improving during iterations.

The validation dataset is commonly used to observed how the model performs on unseen data. When this is the case, testing data should still be used. This is because hyper parameters, as for example the learning rate, of the model are now adjusted to favor a good validation set, and might not genenralize further. However, the training and validation performance scores will likely start diverging at some point, as demonstrated in Figure 6-1. This is usually when we consider the a good stopping point found in the supervised case. As training goes on, overfitting might also cause the validation accuracy to decrease.

6.2. Gradient descent

Even if overfitting was not a problem when obtaining the extrema of f with respect to $\boldsymbol{\theta}$, analytically obtaining the minima of J^* in (2), or even J in (4), would be either impossible or infeasible for most deep learning models. The goal remains unchanged: to obtain a low value of the error function, but also constrained to return satisfactory results for both training and validation dataset.

To iteratively tend towards the minima of $\mathcal{E}$, a very common approach is to use some variation of gradient descent [23]. Gradient descent is a method where the gradients of the error function are estimated, and then used to update the parameters:

$$\theta^{(t+1)} = \theta^{(t)} - \mu \frac{\partial \mathcal{E}}{\partial \theta^{(t)}}, \tag{5}$$

where μ is called the learning rate.

In gradient descent, the parameters are updated scalar-wise. $\boldsymbol{\theta}$ does not *have* to be a vector as it can be any collection of parameters. Here the scalar representation of one parameter at training iteration t is simply denoted $\theta^{(t)}$, and represents any and all parameters in $\boldsymbol{\theta}$ at t.

For smaller architectures, getting stuck in local minima is a major problem [57]. To address this it is very common to apply an algorithmic scheme that is called SGD. Stochastic gradient descent is equivalent to gradient descent, except from the sole difference that the data is randomly sampled (often without replacement) into smaller batches. The loss function is calculated over the batch, and thus the parameters of the architecture is then updated batch-wise as batches propagate through the model. When all the sampled data points have been through the training procedure once, then that is commonly referred to as one epoch.

Stochastic gradient descent performs significantly better than its ‘deterministic’ counterpart. In fact, gradient descent without stochastic mini batching is nearly unheard of in modern literature.

However, as the network size grows, local minima becomes of less of a problem as the optimization algorithms are able to navigate them [57].

Gradient descent, as well as the stochastic version, has trouble navigating saddle points and other areas where the surface curves significantly more steep in one dimension than in another [58, 59]. When the SGD algorithm encounters such a saddle point, SGD oscillates across the slopes of the ravines while only making hesitant progress along the bottom towards the local optimum. Luckily, the literature provides many suggestions as to how the performance can be improved especially through variations within stochastic gradient descent. These techniques are commonly referred to as optimizer algorithms, or just optimizers.

6.3. Momentum and Adam

Momentum [60] is a method that helps accelerate SGD in the relevant direction and dampens oscillations. This is achieved by adding a fraction γ to the update vector of the past time step to the current update vector.

The update rule now becomes

$$\begin{aligned} v^{(t+1)} &= \gamma v_{t-1} + \mu \frac{\partial \mathcal{E}}{\partial \theta^{(t)}} \\ \theta^{(t+1)} &= \theta^{(t)} - v^{(t+1)}. \end{aligned} \tag{6}$$

The Adam optimizer

The optimizers presented in (5) and (6) are considered fairly simplistic compared to state-of-the-art algorithms. The optimizer that is heavily favored in the literature is the adaptive moment estimation (ADAM) [52] optimizer. The ADAM optimizer calculates the two first moments and performs bias correction, with the following update rules [52].

First estimating the moment

$$m^{(t+1)} = \beta_1 m_t + (1 - \beta_1)\frac{\partial \mathcal{E}}{\partial \theta^t} \tag{7}$$

$$v^{(t+1)} = \beta_2 v_t + (1 - \beta_2)\left(\frac{\partial \mathcal{E}}{\partial \theta^t}\right)^2 \tag{8}$$

then performing bias corrections

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$

and finally update the parameters

$$\theta^{(t+1)} = \theta^{(t)} - \mu\frac{\hat{m}_t}{\sqrt{\hat{v}_t}},$$

Even though ADAM is the most popular optimizer, and seemingly the one that often yields the best results, there are overwhelmingly many variations of gradient descent optimizers. The inclined reader is referred to *An Overview of Gradient Descent Optimization Algorithms* by Ruder [58].

Interestingly, the optimizers tend to attain the best results when the momentum terms are high. For instance Kingma and Ba [52], the authors of ADAM, proposes that $\beta_1 = 0.999$ and $\beta_2 = 0.9$, which successively indicates that the overall structure might be more important than the local ones [56, 57].

ADAM has also received some criticism in regards to convergence recently [61]. However, to the authors best knowledge, it is still considered the optimizer that has the best chance of attaining good results, despite moment estimations confusing discriminative adversarial training.

6.4. Unsupervised learning

So far, only the supervised case has been considered. Even though the goal remains pretty much the same, unsupervised learning has an additional constraint. As this thesis will perform experiments that are considered unsupervised, the rudimentary philosophy of unsupervised learning will be presented here.

In supervised learning, a measure of performance, $\mathcal{A}$, is indirectly optimized. This measure directly provides information on how the model is performing by comparing a priori 'end goals' with the output of the architecture. That is, supervised learning depends on 'end goal' values ($\mathbf{y}_i$'s). Datasets containing these values are referred to labelled datasets. However, consider the case when the values, $\mathbf{y}_i$, accompanying our samples, $\mathbf{x}_i \sim \mathbb{p}$, are not to be considered by the model. More specifically that means obtaining $\mathbf{y}_i|\mathbf{x}_i$ is impossible. The lack of labelled data is a big challenge in deep learning, and by definition, supervised methods cannot work with such data.

An *un*supervised architecture is defined as such by *not* utilizing prior knowledge about what the output will be. Hence, an unsupervised performance measure measure that utilizes a priori 'end goals' cannot exist by definition [62]. Nevertheless, in unsupervised learning, there also has to be defined *some* measure to optimize. In fact, Section 12 proposes new performance measure will be presented, and as will be apparent, will be useful in certain unsupervised cases.

In contrast to a supervised loss function, an unsupervised loss function is a function that does not depend on supervised labels in any way. Let $\mathcal{L}$ be said measure, then instead of attempting to obtain the map, $f^* : \mathbf{x}_i \mapsto \mathbf{y}_i$, a different more generic parametric map is sought:

$$g : (\mathbf{x}_i, \boldsymbol{\theta}) \mapsto g(\mathbf{x}_i, \boldsymbol{\theta}),$$

where $\mathcal{L}(g(\mathbf{x}_i, \boldsymbol{\theta}))$ attains the optimal value of $\mathcal{L}$. Again this has to hold for all samples in the distribution.

A loss function in the unsupervised case could be defined accordingly as

$$J^*_{\text{unsup}}(\boldsymbol{\theta}) \equiv \mathbb{E}_{\mathbf{x}\sim\mathbb{p}}\mathcal{L}(g(\mathbf{x}; \boldsymbol{\theta})), \tag{9}$$

where $\mathcal{L}$ is some measure of optimization that only takes the estimated probabilities from data points, and of course should be unaware of their true labels.

Accordingly, with the same arguments as for (2), it cannot be directly attained, and again is estimated as

$$J_{\text{unsup}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \mathcal{L}(g(\mathbf{x}_i; \boldsymbol{\theta})). \tag{10}$$

The typical procedure for obtaining the map g is still performing gradient descent on the parameters $\boldsymbol{\theta}$ such that $\mathcal{L}\left(g(\mathbf{x}; \boldsymbol{\theta})\right)$ is optimized for all the samples $\mathbf{x}$ in the distribution. The unsupervised goal can be formulated as first choosing the architecture, g, and then obtain $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{arg optimize}}\, \mathcal{L}\left(g(\mathbf{x}; \boldsymbol{\theta})\right)\ \forall\, \mathbf{x} \in \mathbb{p}. \tag{11}$$

Clustering is on of the most popular branches of unsupervised learning and would commonly apply some transformation to the data and then define the loss $\mathcal{L}$ such that it reflects how well the data is clustered. A cluster is often thought of as some automatic grouping of data, that reveals some underlying, often semantic, structure. Clustering can be referred to as 'unsupervised segmentation' and 'unsupervised classification' [63], because as in classification adn segmentation, the end goal is to assign instances to classes.

The difference is that said classes are not known to the model a priori, but has to be learned autonomously in the unsupervised case. A cluster could be defined using some input data $\mathbf{X}$ together with some assignment map $\mathcal{C} : \mathbf{X} \mapsto c$ where $c \in \mathsf{C}$ and C is a set containing the possible clusters. $\mathcal{L}$, when clustering, is often a measure of the density and separability of distributions, as for instance the Cauchy Schwartz divergence depicted in Figure 6-2.

Deciding what performance measure to optimize, as well as how to implement the $\mathcal{L}$ function, is very problem dependent. Many such functions are often statistically motivated based on probability distributions.

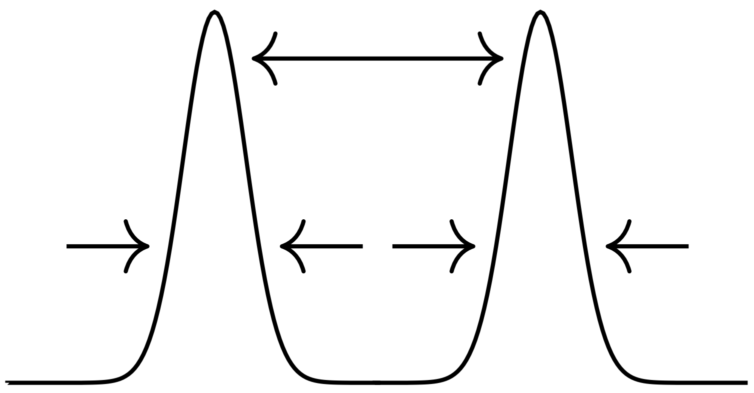


Figure 6-2: *Illustration of characteristics of two distributions with a low Cauchy-Schwarz divergence*

A quick example would be the Cauchy-Swartz divergence, which was used in another one of the authors prior publications *Recurrent Deep Divergence-Based Clustering for Simultaneous Feature Learning and Clustering of Variable Length Time Series* [64], much based on *Deep Divergence-Based Clustering* by Kampffmeyer et al. [65]. The idea with Deep Divergence Based Clustering [65, 64] is that the data is assumed to consist of clusters that can be transformed to a space where they represent, in large parts, linearly separable groups if the data is transformed into a space where the probability distributions are more distinctive. In [64] we apply the Cauchy-Swartz divergence (Figure 6-2) as a measurement of how divergent (different) multiple probability distributions are.

6.5. Loss functions

Loss, or cost, functions are generic functions that are designed to evaluate how well an algorithm is performing. The loss function should be continuous and differentiable for back propagation purposes and, as motivated in Section 6.1, be a monotonic function that has a local minima. If we recall the definition of the loss function from (2), the expectation itself is not attainable, and what is to be presented here are the estimations of the loss function. Here the loss functions are estimations of such (as suggested in (4)).

As will be discussed later, in the experiments of this thesis, the cost function in the unsupervised part is actually the cross entropy, which is primarily used for supervised learning. Consequently, no loss functions that are typical in the unsupervised case will be presented here.

6.5.1. Loss functions in supervised learning

In supervised learning, the cost functions are functions of the label and the output of the architecture, and is often interpretable as a distance measure between the true label and the prediction made by the model. The choice of loss function is not a trivial one, and is very problem dependent. Here, the theory behind the two most common loss functions will be introduced. In these loss functions, consider the true labels of data sample i: $\mathbf{y}_i$ and let $\hat{\mathbf{y}}_i$ continue be the output from the model.

Recall that a loss function is typically minimized, and as such, loss functions are a ordinarily non-convex function that has a local minima.

6.5.2. Mean squared error

The mean squared error is commonly used in regression problems. Especially in linear regression. However, it has in the literature proved useful in some simpler use cases. The 'error function' would be just the squared error

$$\mathcal{E}(y_i, \hat{y}_i)_{\text{mse}} = (y_i - \hat{y}_i)^2$$

And the actual loss function to be used in training (the estimation of (2)) is

$$J_{\text{mse}} = \frac{1}{N}\sum_i^N (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2.$$

6.5.3. Cross entropy

In the experiments presented in Part IV, this is the loss function (with minor modifications due to dimensionality differences) that is used for all architectures. Following the notation, the cross-entropy loss could be expressed as the log-likelihood[c] of $\mathbf{y}_i|\mathbf{x}_i$:

$$H(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \sum_j -y_{j,i}\log(\hat{y}_{j,i})$$

Where $\hat{y}_{i,j}$ is interpreted as a probability (that is $0 < \hat{y}_{i,j} < 1$ and $\sum_j \hat{y}_{i,j} = 1$).

From this, since $y_{i,j} = 1$ only for one value of j this would ultimately only be the logarithm of the probability of the class that the data point actually belongs to. The entropy is thus higher for $y_{i,j}$ closer to 0 and lower for $y_{i,j}$ closer to 1.

The resulting loss function would often be the sum or average of the cross entropies

$$J(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_j\sum_i -y_{j,i}\log(\hat{y}_{j,i}).$$

[c] There are other interpretations of this loss function, and it has a solid base also in information theory.

6.6. Performance measures

When dealing with classification or segmentation, we want to know the classifiers ability to classify correctly, or what here is called performance measures $\mathcal{A}$. Ordinarily, these performance measures will have the common property of making use of a priori information about the end classification goal (labels), and compares it with the output of the model (predictions). This thesis will propose a performance measure in Section 12 that will be further explored in the experiments in Part IV. Here common performance measures will be presented to give the reader an overview and insight in common, and already established, performance measures.

One natural, and commonly used performance measure, is the fraction of data points that were correctly classified (accuracy). However a challenge with this is that in many cases, the positive labels only constitutes a small part of the data points with the negative class being the rest of said data points (or visa versa). For instance, the (relatively small) tumor in a PET image is the positive class whilst all the pixels without a tumor would be negative.

We are still interested in knowing the cases the classifier says are the true (or positive) class. In segmentations considered in the experiments, this a priori knowledge often consist of 'regions of interest'. For the same reason as above, the number of true positives is not very informative per se. Knowing only how many true positives there are does not help solve the problem. There has to be some reference point to make the number of true positives more meaningful. This is where the different performance measures come in handy. It makes much more sense to know, for example, what portion of the positives were correctly identified somehow compared to the fraction of negatives were correctly identified.

6.6.1. Common measures

For classification, especially for two-class problems, variety of measures have been proposed. When doing prediction on a two-class problem, there are four possible cases. These are combinations of what the model predicts compared to what the truth is. For example, a model can predict that a datapoint is 'true', typically 1, but the ground truth is that the datapoint is false, typically 0. In that particular case we would have a false positive [66]. This is further demonstrated in Table 6-1. Different combinations of the confusion matrix form well known performance measures as [66] and an overview is presented in Table 6-2:

	Predicted class		
True class	Positive	Negative	total
Positive	t_p: True positive	f_n: False negative	p
Negative	f_p: False positive	t_n: True negative	n
Total	p'	n'	N

Table 6-1: *Confusion matrix for two-class problem*

Name	Expression	Explanation
error	$\frac{1}{N}(f_p + f_n)$	Fraction of wrongly classified observations
accuracy	$\frac{1}{N}(t_p + t_n) = 1 - \text{error}$	Fraction of correctly classified observations
tp-rate	$t_{pr} = \frac{t_p}{p}$	Also referred to as 'recall', sensitivity and hit rate. Fraction of correctly classified objects that are 'true'.
fp-rate	$f_{pr} = \frac{f_p}{n}$	Also known as the *false alarm*-rate. The fraction of positive classifications that should have been negative.
precission	$\frac{t_p}{p'}$	Fraction of predicted positives that are correctly classified.
specificity	$\frac{t_n}{n} = \text{1-fp-rate}$	Fraction of observations correctly classified as 'negative' or 'false'

Table 6-2: *Common performance measures used in two-class problems*

6.6.2. Dice (F_1) score

The dice [67] score is a measurement of overlap. The images for segmentation often only have a small area with positive scores. If the classifier learns to predict all pixels to the negative class, one would still be able to get a high accuracy, since most of the pixel-wise classification is correct. This despite the fact that the region of interest may not be detected at all. Examples of this will be demonstrated in the experiments (Table 15-5 on page 92).

The dice coefficient is the overlapping area ($A \cap B$) divided by sum of the areas of the prediction and the ground truth ($A + B$). This way a big area of prediction or ground truth will tend to have small dice score values if the overlap between the ground truth and prediction areas are low.

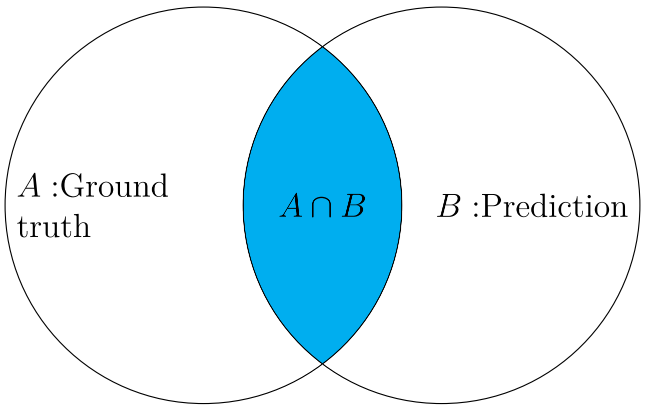


Figure 6-3: *Venn diagram for explaining Dice coefficient*

The dice score is given as

$$\mathcal{A}_D = \frac{2A \cap B}{A + B}. \tag{12}$$

The F_1 score is fundamentally equivalent. However, it is formulated in such a way that it provides a definition that extends beyond overlapping. F_1 will take class imbalance into account and will tend to favor true positives over punishing false positive, as would the Dice. The F_1 score is defined as the harmonic mean of the precision and recall:

$$\begin{aligned}\mathcal{A}_{\mathrm{F}_1} &= 2\left(\frac{1}{\text{precission}} + \frac{1}{\text{recall}}\right)^{-1} = 2\left(\frac{\text{precission}\cdot\text{recall}}{\text{precission}+\text{recall}}\right)\\ &= \frac{2t_p}{2t_p + f_n + f_p}.\end{aligned}$$

Note that $f_n + f_p$ is the total number of wrongly classified observations and t_p the number of true positives.

It's easy to show that the dice score is equivalent to the F_1 score. If we consider an image with A being the area containing the positive ground truth, and B the predictions, as in Figure 6-3. Then $A \cap B$ would be the number actual positive pixels correctly classified as such (t_p). Furthermore, A could be expressed as the overlapping region (t_p) in union with the pixels that should have been classified as positives but weren't: $A = t_p + f_n$. Then B would be the true positives in union with the pixels that were classified as positive, but were in fact negative $B = t_p + f_p$:

$$\mathcal{A}_D = \frac{A\cap B}{A+B} = \frac{2t_p}{2t_p + fn + f_p} = \mathcal{A}_{\mathrm{F}_1},$$

and the F_1 score is identical to the dice score.

6.6.3. Area Under the ROC Curve

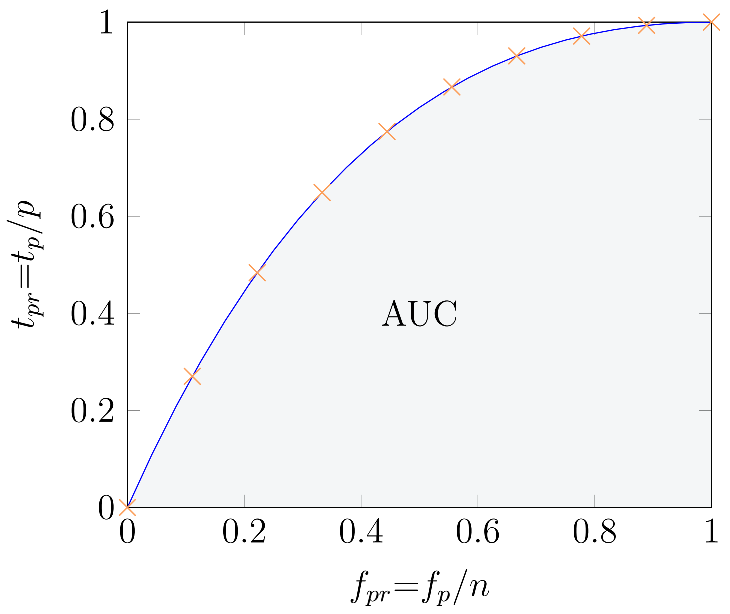


Figure 6-4: *A classifier is often preferred if its ROC curve is closer to the upper-left corner (thus large AUC).*

When performing classification on a two-class problem, we want a measurement telling us how many positive classes did we get right. This is the tp-rate $t_{pr} = t_p/p$. However, if this was our only measurement the classifier could just assign all the data points to the positive class, and similar to the accuracy case, perform an ideal tp-rate: $t_p = 1$. Therefore it is important to also look at the negative classes. If we simultaneously look at the fp-rate $f_{pr} = f_p/n$ we get the ratio of wrongly classified negatives; We simultaneously want a high t_{pr} and a low f_{pr}.

Classification models often returns soft classifications. That is a number between 0 and 1 where, say, 0 is negative and 1 is positive. To be able to calculate our t_{pr} and f_{pr} these soft predictions have to become 'hard'. A common way of doing that is just by thresholding. However, the values of t_{pr} and f_{pr} will depend a lot on the value of said thresholding. For example, if we say that all values greater than 0.001 is positive, it is likely that we get a high t_{pr} but also a high f_{pr}. Similarly if a high threshold (e.g. true if greater than 0.999) then we'd get a low t_{pr} and, likely, a low f_{pr}.

Choosing this threshold is a matter of whether it is favorable to have false positives compared true positives. For instance, this was to help decide whether someone should get a thorough diagnosis on a terminal disease, one would likely be okay with a higher f_{pr} than if it was to decide whether a person should go to prison. In other words, it is a design choice.

The ROC curve allows to model this trade off. AUC (the area under the ROC curve) is also independent of said threshold. The area under the ROC curve thus suggests how much 'freedom' one has to choose said threshold. It can also be viewed as a measurement of how certain the model is in its predictions.

AUC would be a value between 0.5 and 1.0 where 0.5 would be the expected result if the classifier was guessing at random.

7. Neural Networks

The neural network is the heart of deep learning, and might be considered the most fundamental architecture used in the experiments in Part IV of this thesis.

As motivated in the introduction, the neural networks, and their variations, are natural choices when performing segmentation, classification and adversarial discrimination, as it currently holds state-of-the-art status in many of these tasks.

When ADDA is introduced in Section 10.1, it will be clarified that in prior to applying the adversarial discriminative domain adaptation algorithm on images, a source map has to be obtained. In the case of the experiments performed in Part IV, the source map a part of a segmentation scheme. As also extensively presented in Section 1 of the introduction, a natural choice for segmentation, and image classification in general, is the convolutional neural network architectures. convolutional neural networks are special types of neural networks.

First the general feed-forward neural network will be presented, thereafter the so called ‘back propagation’. This is then followed by the convolution operation which motivates the convolutional neural network. CNNs are presented thereafter and is followed by the related back propagation. Lastly, this section the ‘transposed convolution’, with back propagation calculations, is motivated as it is applied in architectures such as u-net, which is a fundamental part of the experiments in Part IV.

7.1. Feed forward neural network

To motivate the theory and notation, the reader is first introduced to simple, specific examples, and then the theory is generalized to the multi-dimensional case, and notation is introduced accordingly.

7.1.1. Perceptron

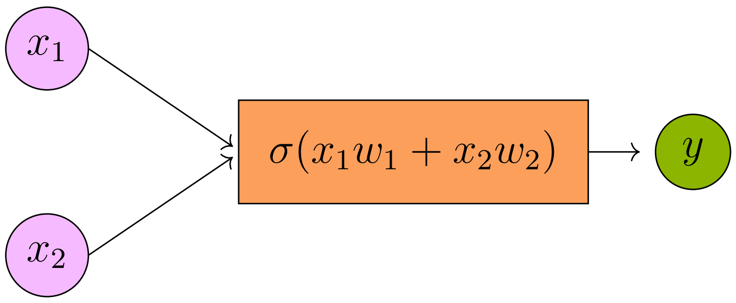


Figure 7-1: *A basic perceptron.*

The simplest neural network is the perceptron. It merely consists of an input layer and an output layer. This simple architecture is essentially an inner product between inputs and parameter weights. This result is often evaluated by a non-linear function (σ in Figure 7-1). The perceptron can also be motivated using logistic regression, and obtain an equivalent result [66].

The output y is often manipulated to be in the range of zero and one so that it can be interpreted as the probability of class affiliation. This is typically achieved by a non-linearity such as the sigmoid function.

To discretize the class affiliation, one could threshold on a value defined in the image of σ, which in turn will divide the plane into two parts which could represent the two classes. Neural networks can in most cases be considered a combination of an arbitrary number of perceptrons, and a feed forward neural network is also often referred to as a multilayer perceptron (MLP).

7.1.2. Forward pass

For clarity, in this two dimensional case, some of the notation will be different in order to be explicit and avoid confusion. First, consider the basic example of the two-layer perceptron in the two dimensional case. We have two values that accompany one 'end goal' value Y_β, which is the 'label'. As earlier motivated, we want to obtain an estimation of $\mathbb{E}[Y_\beta | x_1, x_2]$

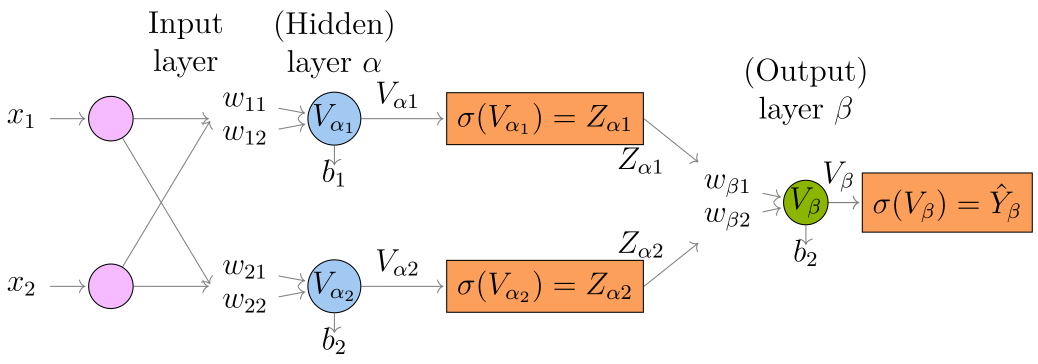


Figure 7-2: *A two layer perceptron with one output layer.*

Figure 7-2 displays, in detail, the different mathematical operations that is performed at each step of this architecture. The first step is to calculate what is shown on the figure as $V_{\alpha i}$s. These are inner products between the parameters and the data:

$$\begin{aligned}
V_{\alpha 1} &= \mathbf{w}_1^\intercal \mathbf{x} + b_1 = w_{11}x_1 + w_{12}x_2 + b_1 \\
V_{\alpha 2} &= \mathbf{w}_2^\intercal \mathbf{x} + b_2 = w_{21}x_1 + w_{22}x_2 + b_2.
\end{aligned}$$

The inner products ($V_{\alpha i}$) are then evaluated in an activation function (similar to what is shown in Figure 7-1), and the resulting quantity is $Z_{\alpha i}$ (i.e. $\sigma(V_{\alpha i}) = Z_{\alpha i}$). This is all now repeated for the next layer to calculate the final output of the network, $\hat{Y}_\beta$:

$$\begin{aligned}
V_\beta &= w_{\beta 1}\sigma(V_{\alpha 1}) + w_{\beta 2}\sigma(V_{\alpha 2}) + b_\beta \\
&= w_{\beta 1}Z_{\alpha 1} + w_{\beta 2}Z_{\alpha 2} + b_\beta \\
\hat{Y}_\beta &= \sigma(V_\beta) = \sigma(w_{\beta 1}Z_{\alpha 1} + w_{\beta 2}Z_{\alpha 2} + b_\beta).
\end{aligned}$$

If the network is expanded, as well as increased in sample size and dimensionality, this notation would be very cumbersome. By generalizing the equation these calculations become much easier:

Assume there are N samples having d observations $\mathbf{X} \in \mathbb{R}^{d \times N}$ and consider a neural network consisting of ℓ layers with n_i nodes in each layer.

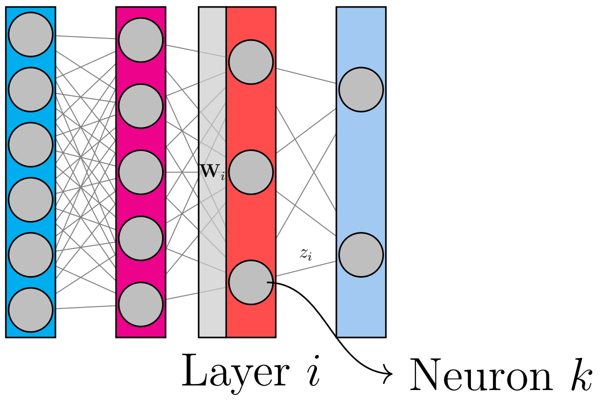


Figure 7-3: *An illustration of a fully connected neural network with notation.*

The entire forward pass can be notationally cleaned up as follows.

Let the parameter weight in layer i of node k is denoted $\mathbf{w}_i^k$. The matrix containing all the weights in layer i is the n_i weight vectors inserted column wise:

$$\mathbf{W}_i = \begin{bmatrix} \mathbf{w}_i^1 & \mathbf{w}_i^2 & \dots & \mathbf{w}_i^{n_i} \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}.$$

Now, let the output of layer i for all N samples be denoted

$$\mathbf{Z}_i = \sigma(\mathbf{V}_i),\ \mathbf{V}_i = \mathbf{W}_i^\intercal \mathbf{Z}_{i-1}. \tag{13}$$

Also note that $\mathbf{Z}_0 = \mathbf{X}$.

To summarize the shapes:

$$\begin{aligned} \mathbf{Z}_i &\in \mathbb{R}^{n_i \times N} \\ \mathbf{W}_i &\in \mathbb{R}^{n_{i-1} \times n_i}. \end{aligned}$$

To demonstrate the role of the loss function, let the last output neuron, $\hat{\mathbf{y}}$, be one single node, such that the squared error can easily be used as the error function. That is $n_\ell = 1$ and $\mathbf{Z}_\ell \in \mathbb{R}^{1\times N}$. Now let $\hat{\mathbf{y}} = \mathbf{Z}^\intercal = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N]$ and y_j be the respective true label for the datapoint j, then

$$J = \frac{1}{2}\sum_j^N (\hat{y}_j - y_j)^2.$$

7.1.3. Back propagation

For the architecture to actually learn something, it is imminent that something is actually updated. Back propagation is the term used when gradient descent is calculating, and applies, the parameter updates to a neural network. The name likely originates from the calculation of gradients in the opposite direction of the forward pass.

Assume n_ℓ neurons in the output layer, then we want to obtain new weights by gradient descent:

$$\mathbf{W}_i^* = \mathbf{W}_i - \mu\frac{\partial\mathcal{E}}{\partial\mathbf{W}_i}, \tag{14}$$

where $\mu\frac{\partial\mathcal{E}}{\partial\mathbf{W}_i}$ works element wise on the weight matrix $\mathbf{W}_i$, and
$\mathcal{E} = \sum_{m=1}^{n_\ell}(y_m - \hat{y}_m)^2$.

Assume that the output of layer i is $\mathbf{z}_i$:

$$\begin{aligned}\mathbf{z}_i &= \sigma(\mathbf{v}_i)\\ \mathbf{v}_i &= \mathbf{W}_i^\intercal \mathbf{z}_{i-1},\end{aligned} \tag{15}$$

and thus $\mathbf{z}_0 = \mathbf{x}$.

We want to obtain, for all weights, the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{w}_r^j}$. By the chain rule in multiple dimensions, we get

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \mathbf{w}_r^j} &= \frac{\partial \mathcal{E}}{\partial \mathbf{z}_{n_L}} \frac{\partial \mathbf{z}_{n_L}}{\partial \mathbf{w}_r^j} \\ &= \frac{\partial \mathcal{E}}{\partial \mathbf{z}_{n_L}} \frac{\partial \mathbf{z}_{n_L}}{\partial \sigma} \frac{\partial \sigma}{\partial \mathbf{v}_{n_L}} \frac{\partial \mathbf{v}_{n_L}}{\partial \mathbf{z}_{n_{L-1}}} \frac{\partial \mathbf{z}_{n_{L-1}}}{\partial \sigma} \frac{\partial \sigma}{\partial \mathbf{v}_{n_{L-1}}} \cdots \frac{\partial \mathbf{v}_r}{\partial \mathbf{w}_r^j},\end{aligned}$$

and from (15) it follows that

$$v_{i,j} = \sum_{\ell} z^{\ell}_{(i-1,j)} w^{j}_{(i,\ell)},$$

and consequently

$$\begin{aligned}\frac{\partial \mathbf{v}_{i,p}}{\partial \mathbf{w}_i^k} &= \mathbf{z}_{i-1} \\ \frac{\partial \mathbf{v}_{i,p}}{\partial \mathbf{z}_i^k} &= \mathbf{w}_{i-1}.\end{aligned}$$

The resulting gradient for the weight now becomes

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}_r^j} = \frac{\partial \mathcal{E}}{\partial \mathbf{z}_{n_L}} \frac{\partial \sigma}{\partial \mathbf{v}_{n_L}} \mathbf{w}^j_{n_L-1} \frac{\partial \sigma}{\partial \mathbf{v}_{n_L-1}} \mathbf{w}^j_{n_L-2} \cdots \mathbf{z}_{r-1}.$$

If we now move over to the dimentionality s.t.

$$\frac{\partial \mathcal{E}}{\partial \mathbf{W}_i} = \begin{bmatrix} \frac{\partial \mathcal{E}}{\partial \mathbf{w}_i^1} & \frac{\partial \mathcal{E}}{\partial \mathbf{w}_i^2} & \cdots & \frac{\partial \mathcal{E}}{\partial \mathbf{w}_i^{n_i}} \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix}.$$

Now define

$$\delta_r^j \equiv \frac{\partial \mathcal{E}}{\partial v_r^j},$$

and the weight update becomes

$$\Delta \mathbf{w}_r^j = -\mu(\delta_r^j \mathbf{y}_{r-1}),$$

where $\delta_r^j = e_{r-1}^j \sigma'(v_{r-1}^j)$, with $e_{r-1}^j = \sum_{k=1}^{n_r} \delta_r^k w_r^{k,j}$.

Moving over to matrix notation, the following matrices are defined for a layer r:

$$\mathbf{V}_r \in \mathbb{R}^{n_r \times N}, \mathbf{Z}_r \in \mathbb{R}^{N \times n_{r-1}+1}, \mathbf{E}_r \in \mathbb{R}^{n_{r-1} \times N}, \mathbf{W}_r \in \mathbb{R}^{n_r \times n_{r-1}+1}, \mathbf{\Delta}_r \in \mathbb{R}^{n_r \times N},$$

containing their respective lower case values. Finally, the updated $\mathbf{W}$ matrix for a layer r would now be

$$\mathbf{W}_{r,\text{new}} = \mathbf{W}_{r,\text{old}} - \mu \mathbf{\Delta}_r \mathbf{Z}_{r-1}, \tag{16}$$

with the following relations:

$$\begin{aligned} \mathbf{\Delta} &= \mathbf{E}_r \odot \sigma'(\mathbf{V}_r) \\ \mathbf{E}_r &= \mathbf{W}_{r+1}^{\intercal} \mathbf{\Delta}_{r+1} \\ \mathbf{V}_r &= \mathbf{W}_r (\mathbf{Z}_{r-1})^{\intercal} \end{aligned}$$

These equations are consistent with what is presented by Theodoridis and Koutroumbas [68].

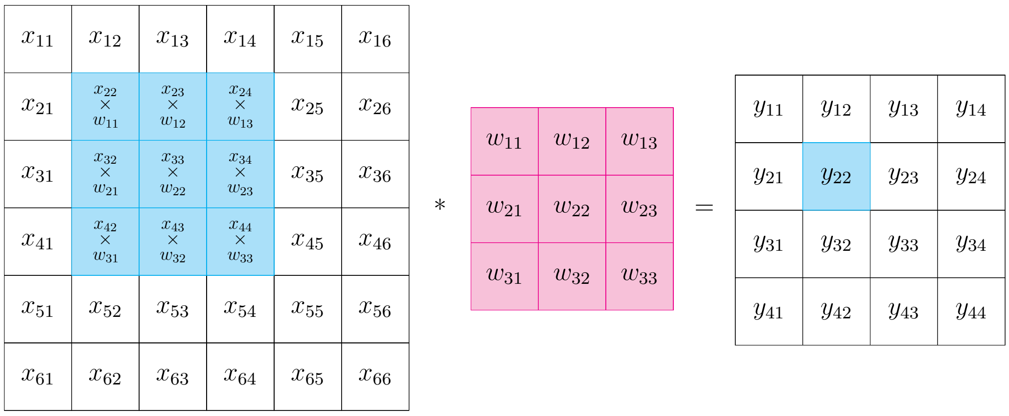
7.2. Convolution

As previously mentioned, the experiments in this thesis rely heavily on convolutional neural networks. This variation of neural networks are based on the convolution operation, as will be introduced here.

For notation, the convolution operator in it's discrete representation in two dimensions is here defined and denoted

$$(X * K)(x, y) = \sum_m \sum_n X(x - m, y - n)K(m, n). \tag{17}$$

Where $X(x, y)$ and $K(x, y)$ will be referred to as the input function and kernel respectively. Note that (17) is commutative and associative.



x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}
x_{21} $x_{22} \times w_{11}$ $x_{23} \times w_{12}$ $x_{24} \times w_{13}$ x_{25} x_{26}
x_{31} $x_{32} \times w_{21}$ $x_{33} \times w_{22}$ $x_{34} \times w_{23}$ x_{35} x_{36}
x_{41} $x_{42} \times w_{31}$ $x_{43} \times w_{32}$ $x_{44} \times w_{33}$ x_{45} x_{46}
x_{51} x_{52} x_{53} x_{54} x_{55} x_{56}
x_{61} x_{62} x_{63} x_{64} x_{65} x_{66}
$*$
w_{11} w_{12} w_{13}
w_{21} w_{22} w_{23}
w_{31} w_{32} w_{33}
$=$
y_{11} y_{12} y_{13} y_{14}
y_{21} y_{22} y_{23} y_{24}
y_{31} y_{32} y_{33} y_{34}
y_{41} y_{42} y_{43} y_{44}

Figure 7-4: *Illustration of the convolution operation*

Convolution is a widely used tool in image processing. Using convolution, one can find edges, find templates and features in an image. Also smoothing and sharpening of images are also common actions to do on an image with convolution as the main tool.

7.3. Convolutional neural networks

It might not be immediately obvious how convolution could improve the performance of a regular deep neural network. convolutional neural networks are commonly known for performing best when applied to classification and segmentation tasks on grid-like structures. For instance in images, videos and text [54]. As it turns out, CNNs have several properties that prove advantageous for solving a multitude of problems.

The most significant distinction between convolutional neural networks and a classical neural network is that the matrix multiplication that constitues the forward pass in (13) is replaced with the convolution operation.

There are four properties that are especially prominent in regards to CNNs [54]: sparse interactions, parameter sharing, equivariant representations, and the possibility of working with inputs of variable size[d]. These properties are the most essential in a CNN as it is foundation in forming the CNNs ability to capture overall semantics in grid-like structures.

In contrast to standard feed forward neural networks, CNNs can have sparse connections between features.

A fully-connected neural network there is one parameter for every value in the input dimension. However, in the convolutional case, the parameters are within the kernels that are often significantly smaller than the number of values in the input dimension. An image, for instance, has millions of pixels. However, convolutional kernels that recognize features such as edge detection could be obtained by small filters with kernel consisting of a single digit number of parameters [54].

Parameter sharing refers to several features sharing the same parameter. That is, for convolution, a parameter contained in the filter is used on every[e] pixel in an image.

[d]There are other architectures that also can do this, e.g. some structures of recurrent neural networks allows for inputs of variable lengths.

[e]Virtually every pixel. Boundary pixels might be excluded depending on the design choices made for the input and kernel (e.g. zero padding).

Because convolutional neural networks typically are applied with significantly fewer parameters than in the classical neural network case, it is naturally considerably less computationally heavy both in memory usage. Together with the skip-connections, CNNs also drastically reduces computational complexity.

In CNNs equivariance relates to position of items. If g is a function that *translates* the input: $g : (x, y) \mapsto (x + \chi, y + \gamma)$, then convolution is equivariant to g: $X * K = g(X) * K$. This is very useful when working with images, as it shows that the convolution operation does not care about *where* a 'feature' is on the image as long as it's represented identically.

We can note that in the sense of how convolutional neural networks treats convolution masks, they could in fact be interpreted as correlations. This is because the networks learns the filters as parameters and thus the high values of the output after the convolution takes place is identical to the estimated cross correlation of the two, as the cross correlation

$$\mathrm{Cor}(x, h) = (X * K)(x, y) = \sum_m \sum_n X(x + m, y + n)K(m, n) \tag{18}$$

The difference from (17) is the sign (in convolution: $X(x - m, y - n)$)

For images, it is often the case that pixels within a local neighbourhood are highly correlated. This is a property that convolution takes great advantage of. When convolution occurs, the resulting output from a pixel will be of high value if the neighbourhood matches many of the pixels. That is if the filter is composed in a way such that it reflects any correlations within the image it will be a high value. For instance if you convolved two photographs together, it would likely have the highest value when the convolution mask is as close to the center as possible.

For example, a filter that has edges characteristic of a human eye would likely show high value of eyes of the same rotation and size as the one of the filter.

7.3.1. Pooling

Convolutional neural networks also often have pooling layers. For instance max pooling and average pooling. In convolution, a window (the kernel) 'slid' across the image. For every iteration (or slide) it performs a weighted sum between the pixels, as shown in (17). The pooling operation, one would replace weighted summation with a different operation for this window. For instance with max pooling the 'cell' that contains the highest value will be chosen, as illustrated in Figure 7-5.

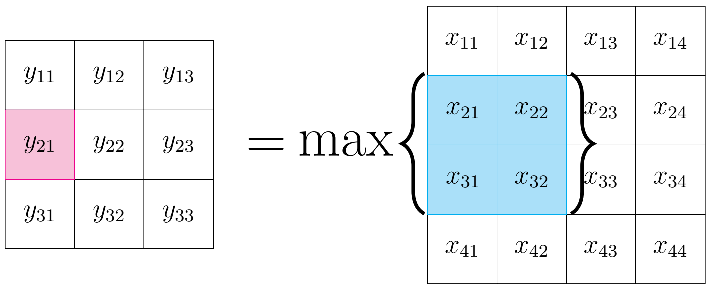


Figure 7-5: *Illustration of the max pooling operation*

7.3.2. Back propagation

Consider the convolution

$$\mathbf{Y}[x,y] = (\mathbf{X} * \mathbf{W})[x,y].$$

For instance the one depicted in Figure 7-4. Here we want to obtain the gradient of the cost function with respect to a weight. By the chain rule we can find

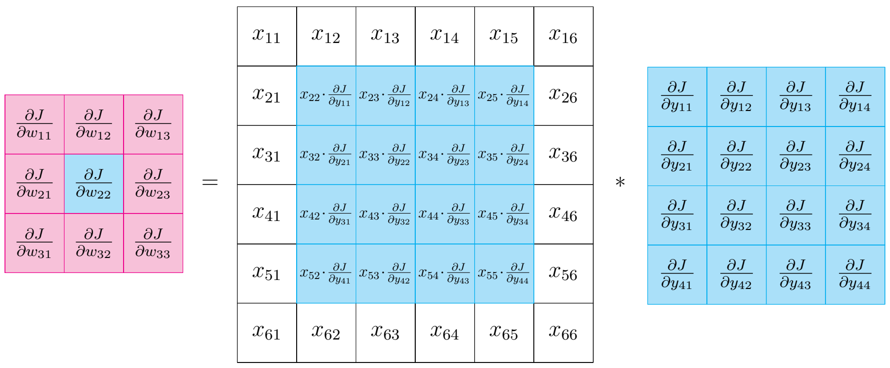
$$\frac{\partial J}{\partial w_{s,t}} = \sum_{i,j} \frac{\partial J}{\partial y_{k,l}} \frac{\partial y_{k,l}}{\partial w_{s,t}} = \sum_{i,j,k,l} x_{i,j} \frac{\partial J}{\partial y_{k,l}}. \tag{19}$$

Of course one have to specifiy the correct limits, and appropriate values for i, j, k and l based on s and t. For instance, in the special case in Figure 7-4, $s = t = 2$ in the limits of the sum would be $i = j \in [2, 5], k = l \in [1, 4]$.

Note here that (19) can be written as a convolution:

$$\nabla_{\mathbf{W}[s,t]} J = (\mathbf{X} * \nabla_{\mathbf{Y}} J)[s,t], \tag{20}$$

where $\nabla_{\mathbf{Y}} J$ is the element-wise gradient $\nabla_{\mathbf{Y}} J[i,j] = \frac{\partial J}{\partial y_{i,j}}$. This is visualized in Figure 7-6.



$\frac{\partial J}{\partial w_{11}}$ $\frac{\partial J}{\partial w_{12}}$ $\frac{\partial J}{\partial w_{13}}$
$\frac{\partial J}{\partial w_{21}}$ $\frac{\partial J}{\partial w_{22}}$ $\frac{\partial J}{\partial w_{23}}$
$\frac{\partial J}{\partial w_{31}}$ $\frac{\partial J}{\partial w_{32}}$ $\frac{\partial J}{\partial w_{33}}$
=
x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}
x_{21} $x_{22}\cdot\frac{\partial J}{\partial y_{11}}$ $x_{23}\cdot\frac{\partial J}{\partial y_{12}}$ $x_{24}\cdot\frac{\partial J}{\partial y_{13}}$ $x_{25}\cdot\frac{\partial J}{\partial y_{14}}$ x_{26}
x_{31} $x_{32}\cdot\frac{\partial J}{\partial y_{21}}$ $x_{33}\cdot\frac{\partial J}{\partial y_{22}}$ $x_{34}\cdot\frac{\partial J}{\partial y_{23}}$ $x_{35}\cdot\frac{\partial J}{\partial y_{24}}$ x_{36}
x_{41} $x_{42}\cdot\frac{\partial J}{\partial y_{31}}$ $x_{43}\cdot\frac{\partial J}{\partial y_{32}}$ $x_{44}\cdot\frac{\partial J}{\partial y_{33}}$ $x_{45}\cdot\frac{\partial J}{\partial y_{34}}$ x_{46}
x_{51} $x_{52}\cdot\frac{\partial J}{\partial y_{41}}$ $x_{53}\cdot\frac{\partial J}{\partial y_{42}}$ $x_{54}\cdot\frac{\partial J}{\partial y_{43}}$ $x_{55}\cdot\frac{\partial J}{\partial y_{44}}$ x_{56}
x_{61} x_{62} x_{63} x_{64} x_{65} x_{66}
*
$\frac{\partial J}{\partial y_{11}}$ $\frac{\partial J}{\partial y_{12}}$ $\frac{\partial J}{\partial y_{13}}$ $\frac{\partial J}{\partial y_{14}}$
$\frac{\partial J}{\partial y_{21}}$ $\frac{\partial J}{\partial y_{22}}$ $\frac{\partial J}{\partial y_{23}}$ $\frac{\partial J}{\partial y_{24}}$
$\frac{\partial J}{\partial y_{31}}$ $\frac{\partial J}{\partial y_{32}}$ $\frac{\partial J}{\partial y_{33}}$ $\frac{\partial J}{\partial y_{34}}$
$\frac{\partial J}{\partial y_{41}}$ $\frac{\partial J}{\partial y_{42}}$ $\frac{\partial J}{\partial y_{43}}$ $\frac{\partial J}{\partial y_{44}}$

Figure 7-6: *Illustration of the convolution gradient backwards propagation*

A more formalized approach for the back propagation, is to consider that convolution *can* be written as a matrix multiplication. Looking at the one dimentional case:

$$(x * h)[t] = \sum_{\tau=1}^{n} x[\tau]h[t-\tau] = \mathbf{x}^{\intercal}\mathbf{h}_t^{*}.$$

Then we can define $\mathbf{x}^{\intercal} = [x[1], x[2], x[3], \ldots, x[n]]$ and $\mathbf{h}_t^{*\intercal} = [h_{t-1}, h_{t-2}, h_{t-3}, \ldots, h_{t-n}]$, and further compose the appropriate matrix required to represent the h function. The resulting matrix for the h function is a variation of the so called Toeplitz matrix:

$$h * x \rightsquigarrow \mathbf{Hx} \begin{bmatrix} h[1] & 0 & \dots & 0 & 0 \\ h[2] & h[1] & \dots & \vdots & \vdots \\ h[3] & h[2] & \dots & 0 & 0 \\ \vdots & h[3] & \dots & h[1] & 0 \\ h[m-1] & \vdots & \dots & h[2] & h[1] \\ h[m] & h[m-1] & \vdots & \vdots & h[2] \\ 0 & h[m] & \dots & h[m-2] & \vdots \\ 0 & 0 & \dots & h[m-1] & h[m-2] \\ \vdots & \vdots & \vdots & h[m] & h[m-1] \\ 0 & 0 & 0 & \dots & h[m] \end{bmatrix} \begin{bmatrix} x[1] \\ x[2] \\ x[3] \\ \vdots \\ x[n] \end{bmatrix} . \tag{21}$$

Here, summing the gradients of the weights (this matrix will result in multiple updates per weight), will result in a back propagation algorithm that is very similar to (16).

For the two dimensional case, one could unravel the data and kernel matrices appropriately to attain a similar result. If we consider the convolution $\mathbf{X} * \mathbf{A}$, with some arbitrary matrix $\mathbf{A}$, then we can rearrange the elements in $\mathbf{X}$ as in Figure 7-7.

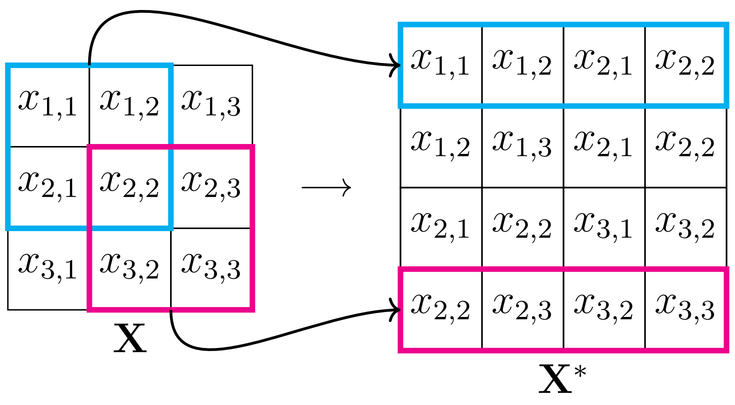

$x_{1,1}$ $x_{1,2}$ $x_{1,3}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$
$\mathbf{X}$
$\longrightarrow$
$x_{1,1}$ $x_{1,2}$ $x_{2,1}$ $x_{2,2}$
$x_{1,2}$ $x_{1,3}$ $x_{2,1}$ $x_{2,2}$
$x_{2,1}$ $x_{2,2}$ $x_{3,1}$ $x_{3,2}$
$x_{2,2}$ $x_{2,3}$ $x_{3,2}$ $x_{3,3}$
$\mathbf{X}^*$

Figure 7-7: *A simplified mapping for 2D convolution*

Then if we do the same unraveling with the $\mathbf{A} \to \mathbf{A}^*$ matrix, then we could write this convolution also as a matrix product:

$$\mathbf{X} * \mathbf{A} \rightsquigarrow \mathbf{X}^{*\mathsf{T}}\mathbf{A}^*.$$

Consequently, calculating the gradient for this layer would be very analogous to the matrix multiplication for the feed forward neural network, by saying that the output of the convolutional layer is the $\mathbf{X}^*$ and the parameters to be trained $\mathbf{A}^*$.

It is important to note that most modern computer machine learning tools and libraries are not using this approach to calculate the gradient during convolution. This representation does, however, justify that CNNs and it's back propagation obey the chain rule.

7.4. Transposed convolution

Several convolutional neural network architectures, especially in segmentation, rely on images to be transformed in the 'opposite direction' of a traditional convolution emerge. For instance, in the aforementioned fCNN [39] architecture, and in the u-net [40] architecture (which will be described in more detail in Section 14.1), up-sampling of the images is a vital part for the architectures.

Transposed convolution, sometimes referred to as deconvolution[f] or fractionally strided convolution, is such a transformation.

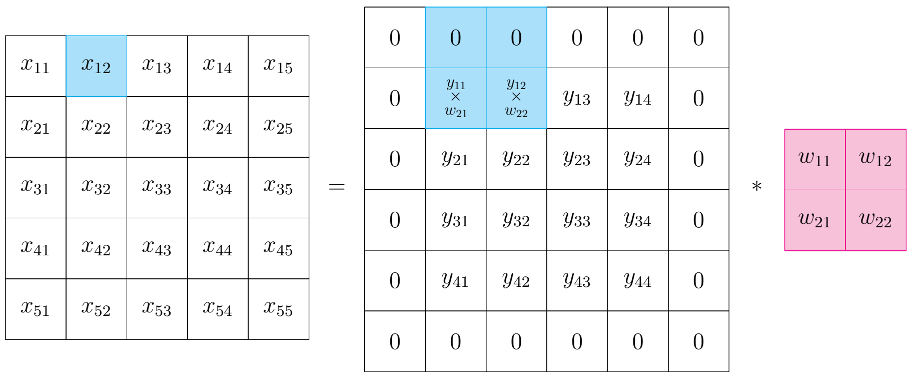
Consider the convolution

$$\mathbf{X} = (\mathbf{Y} * \mathbf{W})[x, y],$$

where $\mathbf{Y}$ has been manipulated such that $\dim(\mathbf{Y}) < \dim(\mathbf{X})$. One way of doing such a manipulation is zero padding, as illustrated in Figure 7-8.

The target objective here is for the model to generate some output $\mathbf{X}$. Let $\mathbf{Y}$ be the input from an earlier layer, and we let the $\mathbf{W}$ be trainable parameters.

[f]Deconvolution might suggest that this transformation is an inverse of convolution. It most definitely is not [69].



x_{11} x_{12} x_{13} x_{14} x_{15}
x_{21} x_{22} x_{23} x_{24} x_{25}
x_{31} x_{32} x_{33} x_{34} x_{35}
x_{41} x_{42} x_{43} x_{44} x_{45}
x_{51} x_{52} x_{53} x_{54} x_{55}
=
0 0 0 0 0 0
0 $y_{11} \times w_{21}$ $y_{12} \times w_{22}$ y_{13} y_{14} 0
0 y_{21} y_{22} y_{23} y_{24} 0
0 y_{31} y_{32} y_{33} y_{34} 0
0 y_{41} y_{42} y_{43} y_{44} 0
0 0 0 0 0 0
*
w_{11} w_{12}
w_{21} w_{22}

Figure 7-8: *Zero padded convolution*

Now, we want to obtain the gradients following the chain rule: $\nabla_{\mathbf{W}[s,t]} J = \sum_{i,j} \frac{\partial J}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{s,t}}$. The gradients with respect to $\mathbf{W}$ is similar to the one shown for normal convolution in (20):

$$\nabla_{\mathbf{W}[s,t]} J = (\mathbf{Y} * \nabla_{\mathbf{X}} J)[s,t],$$

where $\nabla_{\mathbf{X}} J[i,j]$ also is elementwise $\frac{\partial J}{\partial x_{i,j}}$.

It can be shown (rather cumbersomely) that the gradient $\nabla_{\mathbf{X}} J$ is equal to the gradient of input values convolved with a 'flipped' weight matrix $\mathbf{W}^* = \text{flip}(\mathbf{W})$, as illustrated in Figure 7-9:

$$\nabla_{\mathbf{X}} J[s,t] = (\mathbf{W}^* * \nabla_{\mathbf{Y}} J)[s,t].$$

Then we can obtain

$$\nabla_{\mathbf{W}[s,t]} J = (\mathbf{Y} * (\mathbf{W}^* * \nabla_{\mathbf{Y}} J)[s,t])[s,t],$$

and in turn we get, using the associative property of convolution,

$$\nabla_{\mathbf{W}[s,t]} J = (\mathbf{Y} * \mathbf{W}^* * \nabla_{\mathbf{Y}} J)[s,t].$$

$\nabla_{\mathbf{X}} J$ for the example in Figure 7-8 is illustrated in Figure 7-9.

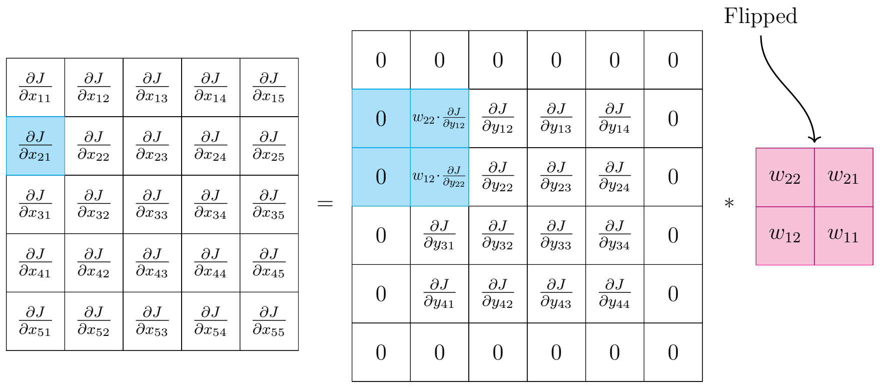


Figure 7-9: *Illustration of the convolution gradient backwards propagation*

Because convolution can be written as a matrix multiplication, we know that these gradients can be computed with the chain rule.

8. Regularization

As discussed earlier, a key challenge in deep learning is making the algorithm perform well on both training and (the withheld) validation data. A diversity of strategies exist that are explicitly designed to reduce the validation error. Such strategies are known as regularization [54]. Regularization is also heavily used in discriminative training [50].

8.0.1. Batch normalization

When a neural network becomes deeper, the small changes within the distributions between layers amplify throughout. When this change in the distributions of these layers occurs, the learning system is said to experience covariant shift [70]. Ioffe and Szegedy [71] argues that this notion can be extended beyond the learning system as a whole to apply to its parts such as sub-network or a layer. They propose normalization via mini-batch statistics to address this, by making a feature of the data points have the mean of zero and the variance of one across the batch. That is for a batch $\mathcal{B} = \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}$ a sample vector $\mathbf{x}_i = (x_1, \ldots, x_m)$ is normalized with $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_m)$, where

$$\hat{x}_i = \frac{x_i - \mathbb{E}_{\mathcal{B}}[x_i]}{\sqrt{\mathrm{Var}_{\mathcal{B}}(x_k)}}.$$

One issue with normalizing each input of a layer, is that it may restrain the input and output when evaluated by a non-linearity (activation function). Another issue is that when propagated through a network, the result might depend on how the data was batched. To address this, Ioffe and Szegedy [71] introduce two trainable parameters γ and β that would be applied to the batch normalization output y_i:

$$y_i = \gamma \hat{x}_i + \beta. \tag{22}$$

During training the batch normalization is applied, but when training is over, only (22) is applied with the trained parameters γ and β.

8.0.2. Weight normalization

If some model parameter, for instance a weight in a neural network, becomes too great of value, it might be indicative of the network learning the dataset and not the overall general structure. The network might be biased against simpler structures and would fail to generalize well. By adding a scaled L2 norm of the parameters as a penalty term in the loss function, you prevent the parameters from becoming big [72].

8.0.3. Dropout

There are many motivations to dropout [73]. One of them is to prevent the model of becoming too dependent on one feature. The idea is to randomly drop units and their connections in a network during training, hoping that it will generalize better. It does this by removing, with a probability p, every unit in a selected layer. To avoid bias, the outputs have to rescaled by multiplying the probabilities with the outputs during validation or testing. Equivalently, divide the output with p during training, which is how tensorflow [74], in which the Part IV experiments are implemented, performs dropout. At test time the resulting neural network should be used without dropout.

9. Adversarial training

Adversarial training is when one architecture improves when 'competing' with the other. As the name suggests, adversarial discriminative domain adaptation is one such architecture, and is part of the main contributions of this thesis. In the following sections, a theoretical background of adversarial discriminative domain adaptation will be presented, starting with introducing generative adversarial networks (Section 9.1), followed by a closer look at the adversarial discriminative training concept (Section 9.2), and is concluded with Section 10.1 introducing ADDA [46] together with performing ADDA on SVHN and MNIST.

9.1. Generative Adversarial Networks

Before diving into the adversarial domain adaptation, it is natural to introduce the source of adversarial training: the generative adversarial network.

GANs could be a clear step in the right direction in regards to process unlabelled data in deep learning, as it is trying to imitate the data distribution, and thus might contribute to dismantle and give some insight to the hidden information within the distribution. Components of GANs are frequently used as intermediate tools to obtain great results in both unsupervised and semisupervised cases.

Fundamentally, a generative adversarial network can be considered a statistical sampler. Generating samples from a distribution given only a population also have many statistical use cases. There are many methods designed for doing this. Today, MCMC and bootstrapping methods are very popular. However, these methods don't generalize well to higher dimensions [75]. GANs, however, have proven very useful for sampling in higher dimensions [49, 50].

In 2014, Goodfellow et al. presented the general adversarial network [53]. The purpose is to generate a sample from a distribution given only a subsample of the data in said distribution. This is achieved by an adversarial 'fight' between two classifiers: the discriminator and the generator.

We continue to let $\mathbf{x}_i$ be a sample from $\mathbb{p}$. Let $\mathbf{z} \in \mathsf{Z}$ and $\boldsymbol{\theta}_\bullet \in \Theta_\bullet$. With slight abuse of notation G here would both represent a domain and a distribution.

Consider the map $G : \mathsf{Z} \times \Theta \to \mathsf{G}$, where $\mathbf{z} \in \mathsf{Z}$ is any (often random) vector, then ideally we want to obtain the surjective map that $\mathsf{G} = \mathbb{p}$, or, $G : (\mathbf{z}; \boldsymbol{\theta}_g) \mapsto \mathbf{x}_i \in \mathbb{p}$.

G here is any continuous differential function $G \in \mathcal{C}^\infty$ with trainable parameters $\boldsymbol{\theta}_g$

Suppose we additionally have a differential function $D \in \mathcal{C}^\infty : (\mathbf{q}_i; \boldsymbol{\theta}_d) \mapsto (0, 1)$, where $\mathbf{q}_i$ is an arbitrary vector and $\boldsymbol{\theta}_d$ are trainable parameters. Then we want $D(\mathbf{q}_i)$ to represent the probability that the sample $\mathbf{q}_i$ originated from $\mathbb{p}$. That is we want to design, ideally,

$$D(\mathbf{q}; \boldsymbol{\theta}_g) = \begin{cases} 1, & \mathbf{q} \in \mathbb{p} \\ 0, & \text{else} \end{cases}$$

Now let $\mathbf{z} \sim \text{Unif}(0, 1)$,[g] then we want to train $G = \arg\max_G D(G(\mathbf{z}))$, that is ideally $D(G(\mathbf{z})) = 1$. That would then imply that $G(\mathbf{z}) \in \mathbb{p}$ for all $\mathbf{z}$.

One possible mathematical representation of this would be

$$G = \arg\min_G (1 - D(G(\mathbf{z}))) \rightsquigarrow \arg\min_G (\ln(1 - D(G(\mathbf{z}))))$$

So D and G are adversaries, where G wants to trick D, whilst D wants to stay true.[h]

[g]Note $\mathbf{z}$ could be chosen in accordance to the probability integral transform, however, the probability integral transform does not generalise easy to higher dimensions. This means we consider $\mathbf{z}$ a prior distribution, and thus $\mathbf{z}$ does not have to be random uniform.

[h]To be consistent with Goodfellow's notation in [53], this part of the thesis will use $\max_g f(g(\mathbf{x}))$ to mean that for functions f and g, is not maximizing the functions, but rather it's parameters: the expression would more correctly be $\max_{\boldsymbol{\theta}_g} f(g(\mathbf{z}; \boldsymbol{\theta}_G))$

We want the discriminator D to correctly label the true data: $\max_D D(\mathbf{x})$. We desire G to maximize the probability $\max_G D(G(\mathbf{z}))$ whilst we simulatneously want D to minimize it $\min_D D(G(\mathbf{z}))$, and we end up with a value function:

$$\min_G \max_D V(D,G) = \mathbb{E}[\ln(D(\mathbf{x}))] + \mathbb{E}[\ln(1 - D(G(\mathbf{z})))]. \tag{23}$$

However, Goodfellow et al. [53] argues that (23) might not provide a sufficient gradient for G to learn well. Rather than training G to minimize $\ln(1 - D(G(\mathbf{z})))$ we train it to maximise $\ln(D(G(\mathbf{z})))$ (or equivalently minimizing $-\ln(D(G(\mathbf{z})))$), which results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning as illustrated in Figure 9-1.

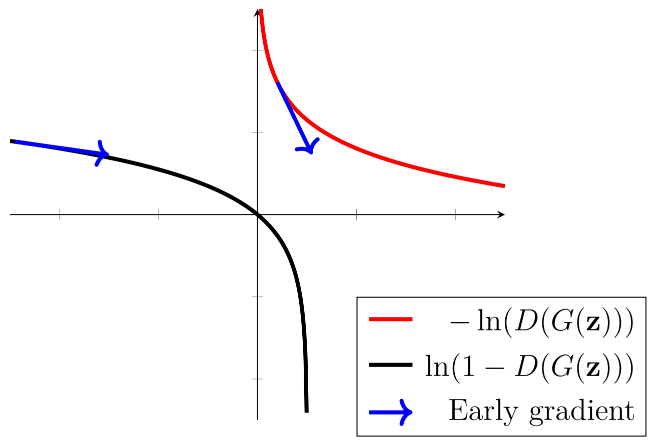


Figure 9-1: *Illustration of difference between minimizing* $\ln(1 - D(G(\mathbf{z})))$ *and* $-\ln(D(G(\mathbf{z})))$.

Optimising D to completion first is computationally prohibitive. Therefore an algorithm that first do k steps training D and then one step of training G is suggested by [53] as follows:

Stepwise GAN training

```
for epochs do
  for $k$ steps do
    // $p_G(\mathbf{z})$ is the (noise) prior of $z$
    Sample $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ from $p_G(\mathbf{z})$
    // $p_{\mathtt{data}}$ is the distribution of the dataset
    // (which is our original data)
    Sample $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ from $p_{\mathtt{data}}(\mathbf{x})$
    // Updating $D(\mathbf{x})$ using SGD
    $\boldsymbol{\theta}_D = \nabla_{\boldsymbol{\theta}_D} \sum_{i=1}^{m} V(G, D; \boldsymbol{\theta}_D, \boldsymbol{\theta}_G)$
  end for
  Sample $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ from $p_G(\mathbf{z})$ // This is a new $\mathbf{Z}$
  $\boldsymbol{\theta}_G = \nabla_{\boldsymbol{\theta}_G} \sum_{i=1}^{m} \ln(1 - D(G(\mathbf{z}_i; \boldsymbol{\theta}_G); \boldsymbol{\theta}_D))$
end for
```

Algorithm 1: *Showing the idea of stepwise iterative GAN training*

These discriminative adversarial training steps are also presented in a slightly more general form in Figure 9-2.

Goodfellow et al. [53] goes on to prove that if G is able to represent the appropriate probability distribution and D is capable of discriminating from it and is allowed to reach its optimum at each step of the algorithm above, the distribution of G converges to the distribution of the data ($\mathsf{G} \rightarrow \mathbb{p}$).

G and D are design choices. Although, they have to be chosen so that they have capacity to represent the discriminant and the original data distribution. Apart from that, the choice of functions does not affect the convergence of the algorithm [53], which allows for a wide range of experimentation.

9.2. Adversarial discriminative training in depth

The GAN is restricted to generative models. However, the idea of adversarial discriminative training can be generalized further. Even the idea of adversarial training alone can be generalized, but that is beyond what will be covered in this thesis.

Similarly to the GAN, the discriminative adversarial training scheme is when one attempts to improve one model by attempting to distinguish (or discriminate) the outputs of this model from another data point.

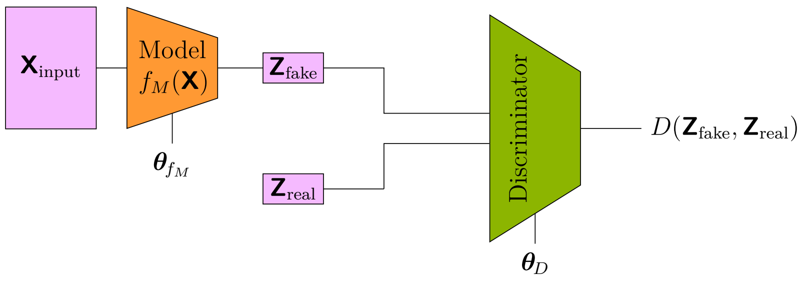
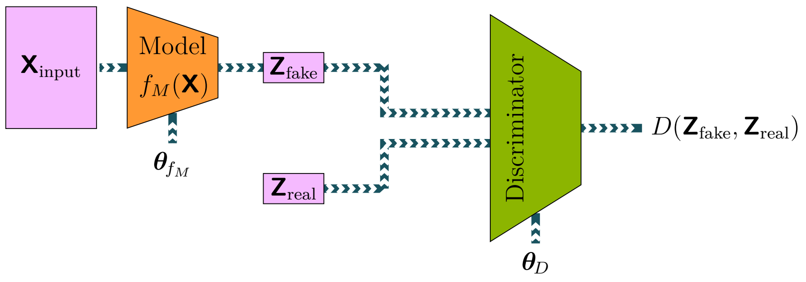

$\mathbf{X}_{\text{input}}$
Model $f_M(\mathbf{X})$
$\boldsymbol{\theta}_{f_M}$
$\mathbf{Z}_{\text{fake}}$
$\mathbf{Z}_{\text{real}}$
Discriminator
$\boldsymbol{\theta}_D$
$D(\mathbf{Z}_{\text{fake}}, \mathbf{Z}_{\text{real}})$

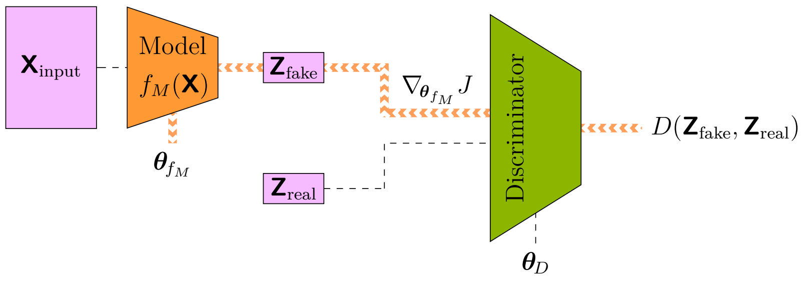
Figure 9-2: *The structure of an adversarial discriminative algorithm.*

Training a discriminator in an adversarial way presents a few challenges. As the name implies, adversarial training can be interpreted as a 'game' between a discriminator and a model. The discriminator is attempting to distinguish between the fake information from the model and the real information from the dataset. Salimans et al. [49] involves a concept called the 'Nash Equilibrium' from game theory into this training procedure. In the case of adversarial discriminative training, a Nash equilibrium can here be interpreted as the solution of this game where both the discriminator and model cannot gain anything by changing only their own 'strategy'. Figure 9-2 shows a fairly generalized setup that introduces discriminative adversarial training.


$\mathbf{X}_{\text{input}}$
Model
$f_M(\mathbf{X})$
$\boldsymbol{\theta}_{f_M}$
$\mathbf{Z}_{\text{fake}}$
$\mathbf{Z}_{\text{real}}$
Discriminator
$\boldsymbol{\theta}_D$
$D(\mathbf{Z}_{\text{fake}}, \mathbf{Z}_{\text{real}})$

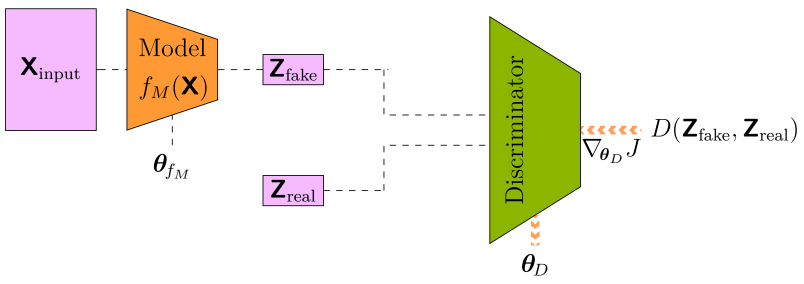
(9-3a) *Step 1: Forward pass.*

The first step is the forward pass. This is when the discriminator outputs a measurement of how well it is able to distinguish between the two data populations: $\mathbf{Z}_{\text{fake}}$ and $\mathbf{Z}_{\text{real}}$.


$\mathbf{X}_{\text{input}}$
Model
$f_M(\mathbf{X})$
$\boldsymbol{\theta}_{f_M}$
$\mathbf{Z}_{\text{fake}}$
$\nabla_{\boldsymbol{\theta}_{f_M}} J$
$\mathbf{Z}_{\text{real}}$
Discriminator
$\boldsymbol{\theta}_D$
$D(\mathbf{Z}_{\text{fake}}, \mathbf{Z}_{\text{real}})$

(9-3b) *Step 2: Back propagate the model.*

The consecutive step is to use the output value from the discriminator to compute gradients, and in turn use the outputs to update the parameters $\boldsymbol{\theta}_M$ ‘in favor of’ the model, whilst $\boldsymbol{\theta}_D$ remains constant.



$\mathbf{X}_{\text{input}}$
Model
$f_M(\mathbf{X})$
$\boldsymbol{\theta}_{f_M}$
$\mathbf{Z}_{\text{fake}}$
$\mathbf{Z}_{\text{real}}$
Discriminator
$\nabla_{\boldsymbol{\theta}_D} J$
$D(\mathbf{Z}_{\text{fake}}, \mathbf{Z}_{\text{real}})$
$\boldsymbol{\theta}_D$

(9-3c) *Step 3: Back propagate the discriminator.*

Figure 9-3: *Adversarial discriminative algorithm steps.*

Finally the gradients for the discriminator parameters $\nabla_{\boldsymbol{\theta}_D} J$ is computed and the parameters of the discriminator, $\boldsymbol{\theta}_D$, are updated. However this is 'in favor of' the discriminator. From this, it is also clear that $\nabla_{\boldsymbol{\theta}_{f_M}} J$ depends on D and $\nabla_{\boldsymbol{\theta}_D} J$ depends on f_M, as is important for the proposed momentum reset scheme. A different note is that despite discriminative training being unsupervised, it is common to use cost functions that are primarily used for supervised learning. Labels in the discriminative case are not from the dataset, but generated entities for discriminating. This will not be in violation with the definition in (10) because it still only depends on the input data. Not any ground truth labels.

More formally, if

$$\mathbf{Z}_{\text{real}} \sim \mathbb{b},\ \mathbf{X}_{\text{input}} \sim \mathbb{p} \text{ and } \mathbf{Z}_{\text{fake}} = f_M(\mathbf{X}_{\text{input}}; \boldsymbol{\theta}_{f_M}), \text{ with } \mathbf{Z}_{\text{fake}} \sim \mathbb{q}, \tag{24}$$

then we want to train the model f_M to align the distribution $\mathbb{q}$ with $\mathbb{p}$ by updating the parameters $\boldsymbol{\theta}_{f_M}$ of the deterministic map f_M. The goal is obtain a map f_M^* that is capable of representing said transform and, with a slight abuse of notation, use this map to obtain the parameters

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{arg optimize}}\, f_M^* : (\mathbf{X}_{\text{input}} \in \mathbb{p}; \boldsymbol{\theta}) \to \mathbf{Z}_{\text{real}} \in \mathbb{q}$$

An interesting observation is that if the underlying distributions $\mathbb{p}$ and $\mathbb{b}$ are too similar, for some definition of similarity, one could argue that the training is not completely unsupervised, even if their labels are known.

In the case where the underlying distributions $\mathbb{p}$ and $\mathbb{b}$ are unequal, is when this type of training can be referred to as unsupervised adversarial discriminative domain adaptation.

In addition to ADDA which will be introduced in the following Section 10.1, there are many different generative models that could be modified to perform unsupervised domain adaptation algorithms.

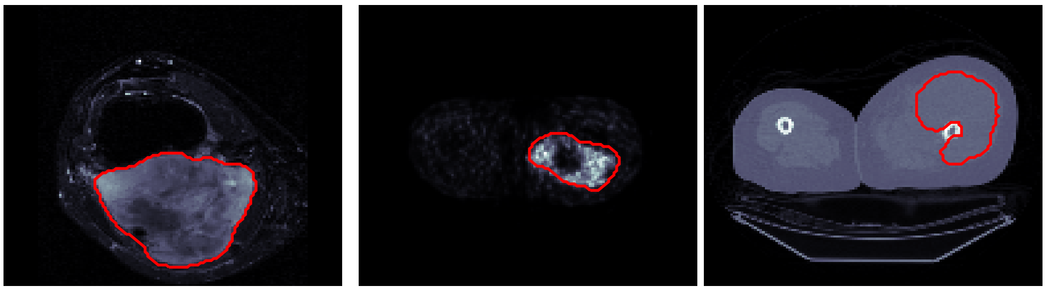
For instance there are variational auto encoders [76, 77], pixelRNN [78] and its convolutional version pixelCNN [79]. All of these architectures have the common characteristic that they provide an explicit parametric specification of a probability distribution (of the input). Such models are known as 'prescribed probabilistic models' [80] or 'explicit density-models'.

In contrast, GANs are models of the category that are referred to as 'implicit (generative) models'. GANs, while still doing probability density estimation as demonstrated in (24), are not restrained to a specific definition for said distribution, and instead define a stochastic procedure that directly generates data [80]. In fairly generic terms, implicit generative models do specify a density distribution in the output space of $\mathbf{Z}_{\text{fake}}$ that in effect forms a likelihood function.

Generally, the key distinction between the explicit and implicit models is that the implicit does not attempt to obtain the parameters to probability distributions, but rather the parameters to the deterministic map (f_M in (24)). Consequently the discriminator, depending on its architecture, is able to learn more complex, and intractable probability distributions compared to if it had to be explicitly defined [81, 80].

Implicit models like GANs are more natural to choose for many problems than its counter part [80]. Because ADDA is heavily based on GANs is a good argument as to why the ADDA scheme is of such importance to explore.

10. Domain adaptation



(a) *MRI (T2)* (b) *PET* (c) *CT*

Figure 10-1: *A PET, CT and MRI (T2) picture of tumours side-by-side. The red lines indicates ground-truth contours of the gross tumor volume (GTV).*

Consider a classifier that is performing excellently on segmenting tumors in a MRI image, such as the one in Figure 10-1a. Now, pertaining to the fact that the tumor is very visible in both the MRI and the PET image (Figure 10-1b), one might think that segmentation of said tumor should be independent of these domains.

Clearly they are different images, and would be even if the tumour was in the same place, but they will in many cases contain overlapping, if not the same, information about a tumor. If we take the MRI classifier under consideration, then the process of tuning it towards understanding PET, images without knowing where in the PET image the tumor is, is unsupervised domain adaptation in a nutshell.

Being clear on terminology, then 'source' data is the data where the ground truth is available. The 'target' data, on the other hand, is the data with inaccessible or withheld ground truth labels, and this is what the model hopefully adapts to being able to recognize in the same way as the target.

Domain adaptation can hence be formalized:

Let $(\mathbf{x}, \mathbf{y})_i \in \mathcal{X} \times \mathcal{Y}$ be a learning sample as before. Assume that its distribution is pairwise such that $(\mathbf{x}_i, \mathbf{y}_i) \sim \mathbb{p}$. Furthermore let $(\mathbf{q}, \mathbf{u})_i \in \mathcal{X} \times \mathcal{Y}$ also be random vectors, but with a distribution $(\mathbf{q}_i, \mathbf{u}_i) \sim \mathbb{q}$. (assuming $\mathbf{u}_i$ exists but are unknown). The goal in unsupervised domain adaptation is to transfer the knowledge of $\mathbb{p}$ and correctly label data from $\mathbb{q}$. That is: to learn a map h such that $h : \mathbb{q} \mapsto \mathbb{p}$ by not utilizing any of the labels $\mathbf{u}_i$. It is here clear that a direct evaluation measurement in a domain adaptation model, that keeps the unsupervised aspect intact, is difficult to define.

In fact, doing adversarial training might be regarded as unreliable due to not having a qualitative performance measure available. It is hard to guarantee that the convergence of the discriminator will necessarily result in a correct and improved measurements of the target map.

Domain adaptation is not to be confused with transfer learning. Transfer learning is a term to describe the fine tuning of a pre-trained classifier. What distinguishes it from domain adaptation is the absence of ground truth, and thus domain adaptation is unsupervised.

Of course, this is a coarse representation, and there is a big variety of methods that can be categorized as domain adaptation. In this thesis the adversarial discriminative domain adaptation model will be introduced in more detail in the following Section 10.1.

10.1. Adversarial Discriminative Domain Adaptation

ADDA [46] is, as the title suggests, a generalized scheme of an architecture that utilizes adversarial training to perform domain adaption. It is a widely used algorithm that is often considered state-of-the-art model, or is the basis of many state-of-the-art models, in unsupervised domain adaptation [46, 48, 33, 82].

In short, the ADDA procedure is obtaining a source map and then to use a discriminator to train, based on the said obtained source map, a target map for a target domain. The source map is often obtained by performing supervised training on a model composed by said source map and a classifier, on a source dataset. Figure 10-2 shows it in more detail.

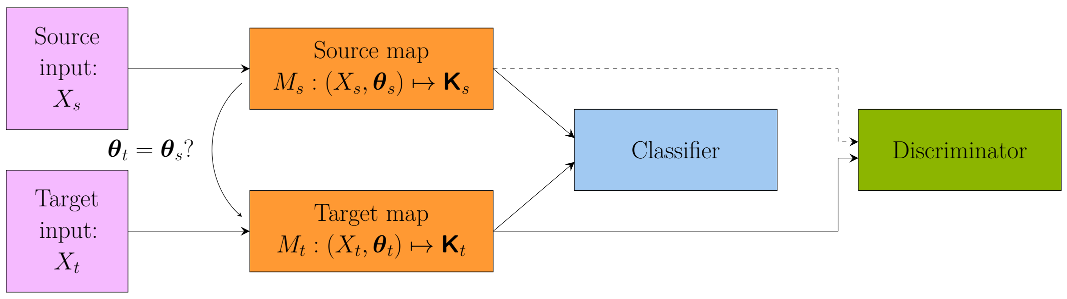
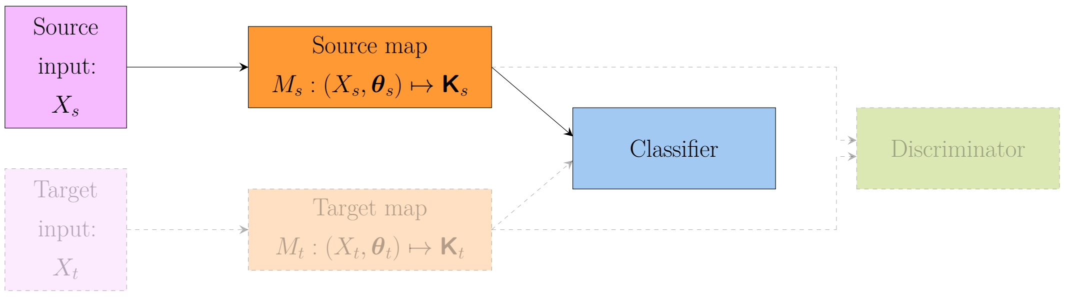

Source input: X_s
Target input: X_t
$\boldsymbol{\theta}_t = \boldsymbol{\theta}_s$?
Source map $M_s : (X_s, \boldsymbol{\theta}_s) \mapsto \mathbf{K}_s$
Target map $M_t : (X_t, \boldsymbol{\theta}_t) \mapsto \mathbf{K}_t$
Classifier
Discriminator

Figure 10-2: *The main ideas behind ADDA in one figure.*

Consider two datasets: a source dataset and a target dataset. For instance, the SVHN and MNIST datasets respectively. The main goal here is to correctly classify MNIST. However, the labels of the MNIST will be unknown to the model, and thus supervised training can only be performed on the source data, SVHN. As this part is presented, SVHN $\rightarrow$ MNIST will be used as example datasets. Results from this domain adaptation, obtained as a part of this thesis, will continuously be reported.

In other words, the target map (MNIST) classification has to be obtained unsupervised. To do this in the ADDA fashion, we are assume that the datasets are somewhat similar in nature, especially in the sense of having similar dimensions and information about what is to be classified.

On a side note, one *could* argue that it is a semi supervised method due to the fact that the source dataset is being trained in a supervised manner. However, the core functionality of ADDA is to align the probability distributions of the two maps, and the classification of the source is not really a part of this algorithm as it is arbitrary. Additionally the literature consistently refers to ADDA as an unsupervised method [46, 48, 33].


Source input: X_s
Source map $M_s : (X_s, \boldsymbol{\theta}_s) \mapsto \mathbf{K}_s$
Classifier
Discriminator
Target input: X_t
Target map $M_t : (X_t, \boldsymbol{\theta}_t) \mapsto \mathbf{K}_t$

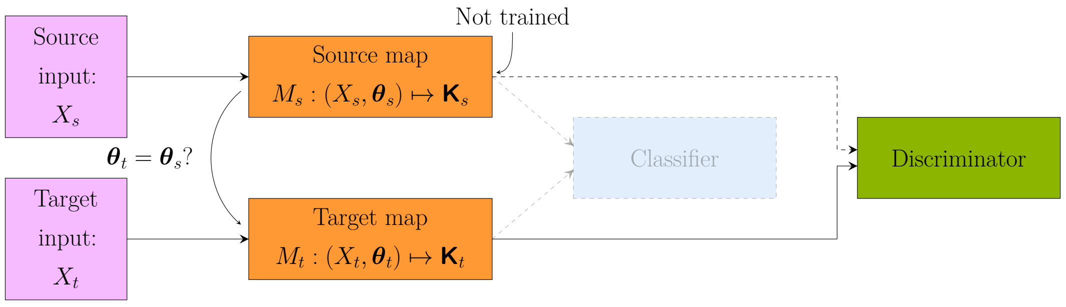
(10-3a) *Step 1: Classify/segment the source data with supervised training.*

Before starting the adversarial training of ADDA (Figure 10-3a) the source map and classifier and has to be trained on the source dataset. The classifier, combined with the source map, would be 'pretrained' end-to-end. That is, just as a 'normal' classifier.

For instance, the SVHN classification was performed using a fairly simple convolutional setup, and provided a **0.9095** classification accuracy.

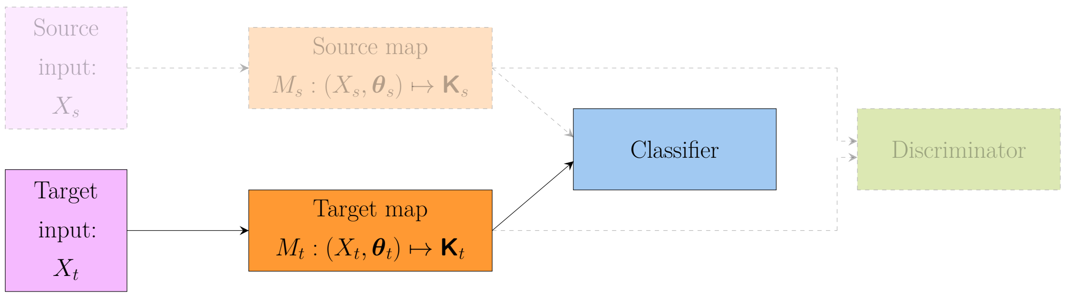
Before the second step, a design choice of parameter sharing has to be made. The choice is whether or not to use a copy of the source mapping parameters (i.e. weights) as initialization for the target mapping parameters $\boldsymbol{\theta}_s$, as seen in Figure 10-3b. Using a copy of the source kernel as the target kernel is a design choice, but in the case of multimodal images, it might be reasonable to assume that the source mapping and target mapping have similar underlying distributions to some degree. Thus it might be fair to expect that the remaining work of aligning the distributions might be merely 'fine tuning' of the copy of the source map.

In the case of SVHN $\rightarrow$ MNIST, the parameters from training SVHN were used to initialize MNIST. When MNIST was propagated at this state, the classification had an accuracy of **0.6424**.


Not trained
Source input: X_s
Source map
$M_s : (X_s, \boldsymbol{\theta}_s) \mapsto \mathbf{K}_s$
$\boldsymbol{\theta}_t = \boldsymbol{\theta}_s$?
Classifier
Discriminator
Target input: X_t
Target map
$M_t : (X_t, \boldsymbol{\theta}_t) \mapsto \mathbf{K}_t$

(10-3b) *Step 2: Adverary training between target map and discriminator.*

When the source map is trained (often supervised) to satisfaction, then the consecutive step is perform unsupervised training of the target domain. This is accomplished using a discriminator that is trying to differentiate whether or not the data originates from the source map. In this step Figure 10-3b, the discriminator is applied to perform adversarial training with the target map. This can be done in many different ways. A popular way is to train the adversarials step-by-step and adjust the number of steps each one of the adversaries should iterate. Notice that neither the source mapping nor the classifier are trained in this step. That is we train the MNIST (source map), and leave the SVHN (source map) and classifier untouched.


Source input: X_s
Source map
$M_s : (X_s, \boldsymbol{\theta}_s) \mapsto \mathbf{K}_s$
Classifier
Discriminator
Target input: X_t
Target map
$M_t : (X_t, \boldsymbol{\theta}_t) \mapsto \mathbf{K}_t$

(10-3c) *Step 3: Classify using classifier*

Figure 10-3: *Adversarial Discriminative Domain Adaptation steps.*

The hope is that, when the adversarial training of the discriminator and target map is completed, the representation of the target mapping of target data is as equivalent to the source mapping of the source input as possible. Section 10.1.1 contains is a summary of and discussion of the the SVHN $\to$ MNIST domain adaptation results.

What is not included in Figure 10-2, but is presented in the generalized scheme from Tzeng et al. [46], is that this model could be used for generative purposes (by making the mappings generators) and also the discriminator here could be two discriminators that are unique for each map.

This could be configured in such a way that the source and target mappings have outputs in the same space. In this case they can be mapped into the ‘segment space. This could resemble segmentation using an auto encoder strategy where the hidden space is our ‘target map’. This illustrates that the it’s hard to provide an unambiguous definition that can provide a clear distinction between domain adaptation and image-to-image translation, especially in the case of image segmentation applications.

10.1.1. Remarks on the SVHN → MNIST results

Table 10-3: *Overview of accuracy scores for SVHN → MNIST domain adaptation. Underline shows best ‘architecture’ while bold numbers show best stopping method (whole row). Asterisk (*) denotes that the results in current scheme was stopped at the same step. Note that there is a difference between* 0.7994 *and* 0.7944

Classification accuracy	Before DA	Scheme	Highest peak	Confusion stop	Latest stop	Best loss stop
0.9095	0.6424	ADDA	0.8218	0.7994*	**0.8128**	0.7994*
		RESET	0.8169	0.7944*	**0.8114**	0.7944*

In this specific case, table Table 10-3 shows that it is not much difference between the proposed methods (confusion score and momentum reset) and the conventional domain adaptation methods.

Momentum reset (RESET) apparently tends to perform a little worse than conventional domain adaptation (DA) on this exact problem. Both SVHN and MNIST are fairly large datasets, with small, clear semantic structures. When the datasets are this big, it might be sensible to argue that the images contained in said datasets represents their probability distributions fairly well. Especially in the MNIST case. This in combination with how similar the images are information wise (all centered numbers), a little the fine tuning might be what was required to attain a fair result, as above. As a consequence, their adversarial loss landscapes might look more similar and the estimated moments might not need to be reset.

Part III / Novelties

Now, that the required background theory is introduced, the reasoning behind the hypotheses stated in the introduction will be further expanded and explored.

When doing adversarial training, each adversarial 'player' attempts to minimize its own cost function, $J_D(\boldsymbol{\theta}_D, \boldsymbol{\theta}_M)$ and $J_M(\boldsymbol{\theta}_D, \boldsymbol{\theta}_M)$ for the discriminator and model respectively. A Nash equilibrium will here represent a point $(\boldsymbol{\theta}_D^*, \boldsymbol{\theta}_M^*)$ such that both cost functions are at a minimum with respect to the parameter of its adversarial [49]:

$$(\boldsymbol{\theta}_D^*, \boldsymbol{\theta}_M^*) = (\arg\min_{\theta_D} J_M, \arg\min_{\theta_M} J_D).$$

As Salimans et al. [49] points out, finding such Nash equilibria is a very difficult problem. Especially in regards to the 'adversarial game' with non-convex cost functions. The problem resides in that, a modification to $\boldsymbol{\theta}_D$ that reduces J_D might increase J_M.

This reveals two clear issues. Firstly, this causes gradient descent to fail, and will prohibit it to converge to the proper point in many games. As a consequence, depending on the loss function alone to stop training will be unreliable. For example if one player is reducing xy with respect to x and the other player reduces $-xy$ with respect to y, then using gradient descent results in a stable orbit, rather than reach the desired equilibrium point $x = y = 0$ [49, 54].

The second issue is that this cross-interaction between the losses might cause problems with the accumulated parameters in the optimizers. Often these are motivated as moment estimations and referred to as 'momentum terms'. Even though using moment based optimizers have yielded good results for adversarial models in the past, it seems to be limited to either generative models or simpler datasets.

In the literature, adversarial discriminative domain adaptation [46] is a fairly young approach with far from overwhelmingly good results, as with many unseasoned unsupervised methods, has been applied to very few real world problems that provided results in the same range as supervised training. Most of the literature presents results either from ‘simple’ datasets like MNIST, USPS or SVHN. However, ADDA is to the author’s best knowledge considered state-of-the-art in unsupervised domain adaptation and, as will be seen in Section 10.1.1, on said ‘simple’ datasets it performs pretty well.

11. Momentum reset

The aforementioned problems bringst to light a very important issue with unsupervised adversarial discriminative domain adaptation. When doing training, moment estimating optimizers are often used, as for example ADAM. Because J_M depends on $\boldsymbol{\theta}_D$ and J_D in $\boldsymbol{\theta}_M$, and the momentum does not change with these cross-interactions between J_D and J_M, the momentum might end up pointing in the wrong direction, as illustrated in Figure 11-1. In fact Radford et al. [50] presents guidelines to make DCGANs stable, where one of the criteria is to reduce the momentum term in ADAM to 0.5 instead of it's original suggested 0.9 [52].

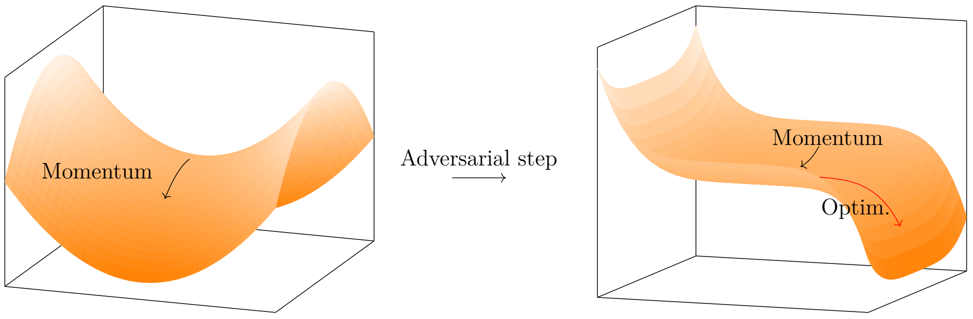


Figure 11-1: *Illustrates the effect when one adversarial updates, the cost function of the other changes while the momentum remains unchanged.*

The issue with a too large momentum rate in the wrong direction is that if the model (the discriminator and/or target map) takes too many steps in the wrong direction, it is more likely that the distribution learned by both the discriminator and the target map would be less semantically correct and tend to more different and random distributions.

One could try to not use any momentum, and only stochastic gradient descent, but then the benefits of momentum [83] would be lost all together.

To still retain momentum, the idea is to let momentum accumulate, and then reset it at a certain point. This reset point could be either after adversarial step training is done, or after a set amount of steps.

This might also allow the discriminative training adversarials to train more steps per epoch iteration, allowing the discriminator to become stronger faster, which might lead to quicker convergence.

12. Convergence

Deciding when a discriminative adversarial model has attained satisfactory results is far from a trivial task. Due to the constant shift in landscape of the loss function, a typical loss in the the adversarial case does not necessarily tend downwards as common loss functions do. In fact, an increase of the loss functions on a mapping adversarial, or generative architecture in an adversarial model, could be considered preferable because it would be consistent with the discriminator doing a better job. This, in turn, might cause the generative adversarial to perform better in its task.

Guaranteeing convergence to a satisfactory, ‘meaningful’, optimum is still an unsolved problem [49, 63]. This makes it very challenging to determine an objective and ‘reliable’ measurement of quality. Such a measure is necessary to determine when the model has reached its optimum, peak performance, and not remain unaware of the best results. For such a stopping criterion to be reliable in regards to the unsupervised domain adaptation, it would ideally have the property of being correlated with a high performance metric (if such a metric exists). Of course, the model cannot know the performance metric due to withheld labels, and thus actually checking correlation would be a thesis in and of it’s own.

For the process to remain completely unsupervised, this criterion must be completely independent of any a priori class information in any performance metrics.

In the original GAN proposal [53], it is shown that after a sufficient number of training steps, the generator distribution, G, will be equal in distribution to $\mathbb{p}$. At this point neither can really improve. When this ‘equilibrium’ occurs, the discriminator would be unable to differentiate between the to distributions: i.e. continuously return values close to $\frac{1}{2}$.

As a criterion for determining convergence of the GAN, just looking at the tendency for the discriminator output value to approach $\frac{1}{2}$ is a bit weak. For instance, if the discriminator classifies all the inputs to positive class, it would output a mean of $\frac{1}{2}$. This in turn does not even hold if the datasets provided to the discriminator are unbalanced.

A more robust requirement could be that the false positive rate and the true positive rate are as equal as possible over multiple steps. That is: following the notation in the confusion matrix in table 6-1: t_p are number of true positives, t_n number of true negatives. p and n are respectively the number of positive and negatives. Then we want to stop the training at step n_s where the recall$= t_p/p$ and the specificity$= f_p/n$ are simultaneously as close to $\frac{1}{2}$ as possible over multiple steps. Specifically, one would want to obtain some measurement of how confused the discriminator is. By formalizing this idea the confusion score is proposed:

$$\mathcal{C}_s^* = \mathbb{E}\left[1 - \left|\frac{t_p}{p} - \frac{1}{2}\right| - \left|\frac{f_p}{n} - \frac{1}{2}\right|\right], \tag{25}$$

or estimated as

$$\mathcal{C}_s = 1 - \frac{1}{N_s} \sum_{k=n_s-N_s}^{n_s} \left(|\mathsf{recall}_k - \frac{1}{2}| + |\mathsf{specificity}_k - \frac{1}{2}|\right), \tag{26}$$

where recall_k and $\mathsf{specificity}_k$ are their respective values for the discriminator at step k.

More specifically this score can be utilized to obtain the iteration n_s^* at which the $\mathcal{C}_s$ was highest:

$$n_s^* = \arg\max_{n_s} \mathcal{C}_s.$$

More specifically implemented similarly to the following pseudo code:

Confusion score implementation

```
best_value=$\infty$
best_value_epoch=0
recalls=[]
specificities=[]
$N_s$=100 // Set some ‘window’ value for determining stopping
for epoch_number=1,...,num_epochs do
  // Train the adversarials
  train_target($n_t$ steps)
  // Train discriminator and get the recall and specificity
  for i=1,...,discriminator_steps
    (recall,specificity)=train_discriminator($n_d$ steps)
    append recall to recalls
    append specificity to specificities
  end for
  if (length(recalls) and length(specificities)) $\geq N_s$ then
    confusion_score=$1 - \frac{1}{N_s} \sum_{k=n_s-N_s}^{n_s} |\texttt{recalls[k]} - 0.5| + |\texttt{specificties[k]} - 0.5|$
    if confusion_score > best_value then
      best_value=test_value
      best_value_epoch=epoch_num
    end if
  end if
end for
final_result=restore_epoch(best_value_epoch)
```

Algorithm 2: *Algorithm that determines discriminator ‘convergence’*

Of course, expecting this method to guarantee an optimal result is not realistic. However, it does yield a performance that might be considered a satisfactory rule of thumb and, as will be shown in the experiments in Part IV, it will in many cases be preferable over the lowest loss or latest training.

This measure of certainty, although designed for unsupervised adversarial discrimination, can have clear use cases in other areas. For instance in supervised models, with a measurement of certainty.

There is a notable property in this regard. A classifier in two class problem can't, for most common performance measures, perform worse than random. Thus given a balanced dataset, a classification accuracy of less than 0.5 have to mean that the classifier is able to classify the data better than random. One notable property is that the confusion score is symmetric in this matter. If f_p is high and t_p is low, simultaneously, it will still give a low confusion score, as by design.

Part IV / Experiments

13. Background

This section will provide background information about the process of 'sanitizing' the dataset to a flexible manageable format. Furthermore a technical outline of how the resulting process works, as well as implementation details in regards to dataset splits and preprocessing.

13.1. Data processing

The dataset is from the cancer imaging archive [84, 85]. It consists of images of sarcoma tumors from three modalities, MRI, PET and CT. Additionally, MRI is divided into two more sub groups: T1 and T2, however only T2 will be considered in this dataset as T1 not provided any satisfactory results and is not considered in current state-of-the-art [2] on segmentation. The data is represented through DICOM (or Digital Imaging and Communications in Medicine) files [86]. DICOM consists of a lot of data: it can contain everything from patient information, image modality, time the photo is taken and so forth. Further more, the ground truth segmentations were in separate files, which was not straight forward to connect to its correct segmentation part. Other issues presented themselves when there were strong inconsistencies in the internal formats in regards to how the tumors and images were stored, and link connections between tumor images and the contour images. Formatting data to a manageable format is of course expected to be a significant part of the workload. However, converting DICOM into a more manageable format proved to require a deeper knowledge of said format then first anticipated. As indicated above, processing said dataset has in this project been far from a trivial task, and properly formatting this data has been a pretty time consuming process. The challenge here was not only to make the data compatible, but also flexible to work with across these modalities. Furthermore, having searchable data, with a consistent train, test and validation split, is important to get reproducible results. Also having the data in searchable format and in a way that is easy to use for machine learning programmatically is very beneficiary.

13.2. The data handling process

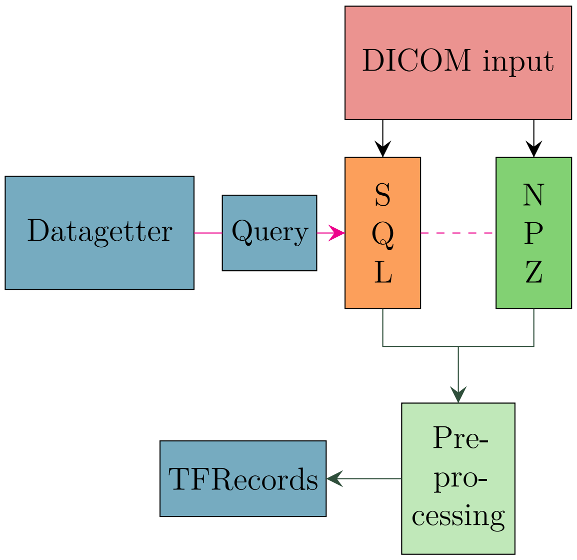


Figure 13-1: *Simplified schematic of the data formatting process*

Presumably, the DICOM format is very handy for medical professionals. However, extracting data into a tidy, consistent format did prove challenging due to differences within each file, study and modality. It is immensely important that the relations between tumor locations and area of scans are properly handled to ensure correct classification and correct reporting of the final results.

This challenge was solved by creating a script that extracted the DICOM information and inserted it into an SQL database with features and a reference to the images that where still stored on disk in Numpy (`.npz`) format. From there the data would be chosen based on queries to the SQL database and converted to TFRecords [87] so that it could be used in training efficiently. The queries selected modalities, part of body that has the tumor, age and other information.

As preprocessing, images were scaled down to 128×128 pixels, using the nearest-neighbour interpolation. Further normalized with the linear scaling minmax $f_{\text{minmax}}(x) = {}^{(x-\min(x))}/_{(\max x - \min(x))}$ for all the pixels in the input to be in the range 0 to 1.

13.3. Dataset splits and preprocessing

The data in the database is easily accessed and flexible based on queries. For example one can simply query to get patients with a tumor in their thigh. To be able to compare results, it is beneficial if the train, validation and test images remained in their respective ‘split’ regardless of what query is performed. This way one would to some degree ensure that good results from one query is not due to the random selection of data points in to train/validation and test splits.

Therefore, every image in the database was assigned to a ‘split’ (either train, test or validation) and remained that way across different queries. This way of splitting the datasets would naturally come at the the cost of not having the exact same sizes across said data selections. It is possible, for example, that the ratio of images in test on validation sets are different based on which age was included in the data selection process.

Another thing to consider when splitting this particular dataset is how to split between patients. For the model to generalize well, it is important that images of certain patients are withheld from training. In this regard, the validation set and the test set contains the same 7 patients (with different images), whilst the training dataset contains images from the remaining 42 patients.

14. Method and setup

Here the specific architectures will be presented. The reader should be aware that the momentum reset scheme is often here denoted as 'RESET', as to make it easier to read. The experiment was written in python 3.6.8 [88] using Tensorflow version 1.13.1 [89].

14.1. The U-Net architecture

U-net was presented in 2015 by Ronneberger et al. [40] with the aim to replace window-sliding patch-wise approaches like Ciresan et al. [90]. X-net is considered one of the very best topologies (architectures) for working with semantic segmentation of medical images [11]. It is clearly a natural architecture to consider in regards to segmentation of medical images from PET and MR modalities.

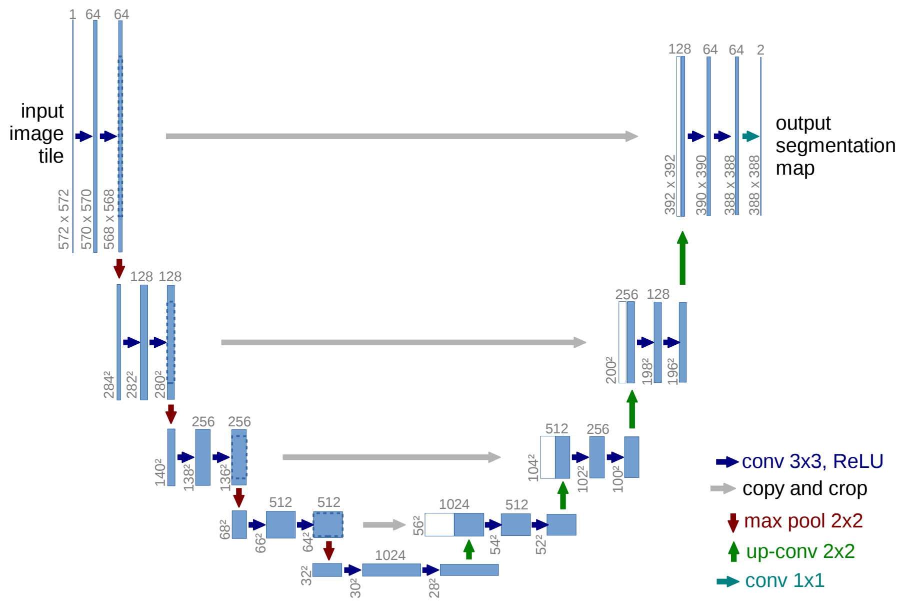


Figure 14-1: *Architecture of original u-net [40].*

The u-net architecture is illustrated in Figure 14-1. It consist of two ‘paths’: one ‘contracting’ path and one ‘expanding’ path. In the original u-net formulation, the contracting part was set up as a very typical convolutional neural network structure [40, 91, 23]: valid convolutions followed by a Rectified Linear Unit (ReLU) [92] and a 2×2 max pooling operation with stride 2 for downsampling, which halves the dimensionality of the previous layer. Before the max pooling, however, a copy of the output is saved for later use.

When the ‘contraction’ path is through, the expanding part begins. Now, the network is upsampled layerwise by transposed convolution. However, at every layer, the aforementioned copy of output at the corresponding ‘level’ is cropped and concatenated with the transposed convolution (see Figure 14-1).

Ronneberger et al. [40] also proposes their own loss function which is essentially an extension of the cross entropy loss, where the pixels are weighted based on their border separations. U-net was originally designed to handle images of a 572×572 input size, and for computational budgeting, they had a batch-size of one.

14.2. Segmentation setup

U-net, based on its remarkable earlier performance on medical images, presented itself as the natural choice of architecture for segmenting the images from the modalities in the dataset. Segmentation was successfully performed using the aforementioned u-net [40].

For purposes of comparing performance on the structure, the same u-net architecture has been used to train all modalities, both in the segmentation and domain adaptation results.

All reported results might further be enhanced if the structure was adapted more to the individual modality requirements.

The current state-of-the-art method for segmenting the soft-tissue sarcoma dataset is *Deep Learning-Based Image Segmentation on Multimodal Medical*

Imaging Guo et al. [2]. Their paper propose a fusion scheme for the images of the tumors together and performs segmentation, in addition to segmenting the modalities one-by-one.

Their preprocessing and data split approaches are similar as to the ones mentioned in Section 13.3, however they only withheld 5 patients for validation compared to the 7 in the thesis experiments.

Their approach consists of dividing every pixel in a given input image into patches of size 28×28. Each of these patches are representative for one pixel, and as such, every patch is classified. The interpretation of the patch classification is then whether or not the pixel, represented by the patch, is a part of a tumor or not. Guo et al. [2] first applies a convolutional neural network and then a fully connected network. Their CNN consist of five convolutional convolutional layers and the fully connected has three layers. More details illustrated in Figure 14-2.

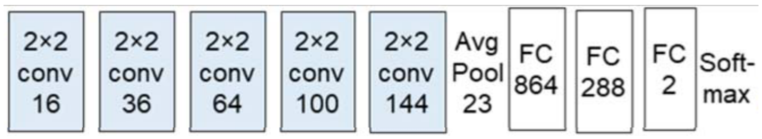


Figure 14-2: *Illustration of the structure from the (prior) state-of-the-art. Taken from Fig.2. in Guo et al. [2]*

In their work Guo et al. [93] it is mentioned that u-net is attempted for segmentation, and yielded similar results.

Even though adversarial discriminative domain adaptation is the main focus of this thesis, segmenting the images is a necessary step to perform the algorithm. It is also reasonable to argue that the quality of result from the ADDA depends on the performed pre-training of the source map, here, segmentation.

MR imaging sequences consists of two imaging sequences per patient: namely T1 and T2. T1 did not perform well and in the experiments, only the T2 feature from the MR images have been used.

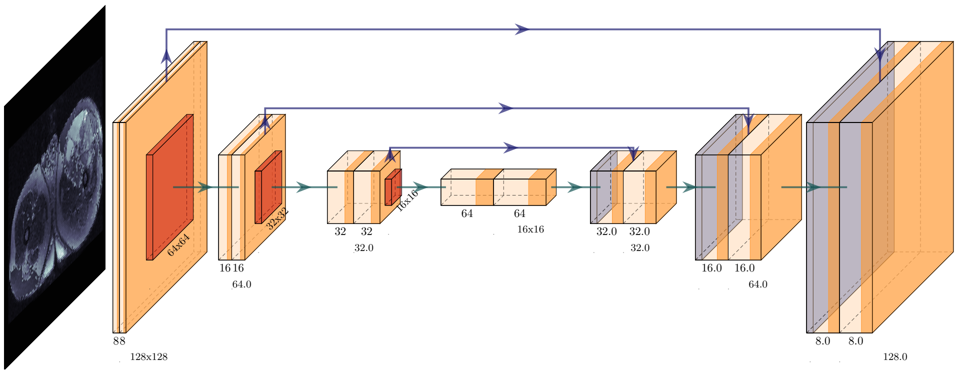


Figure 14-3: *Experiment u-net setup. The beige boxes are convolutional layers are convolutions with rectified linear unit (ReLU) (represented as a darker shade of beige). The orange ones are strided convolutions that will make the size half the input. The grey boxes are upsampling by zero padding. There is batch normalization between every layer.*

Figure 14-3 displays the architecture used for segmentation in this thesis. It is heavily based on the u-net [40] architecture. However, some modifications have been made.

The weighted softmax cross-entropy loss proposed in the original paper, did not indicate to improve the results in the experiments done, and has not been applied. In the original u-net formulation, the U-net had only 'valid' convolutions. In the following experiments 'same' convolution is used. Occurrences of tumors in the edges of the images are degenerate cases, and the 'same' convolution should in principle yield equivalent results. The intention is to allow easier experimentation of depth and input dimensionality. Furthermore, as will be discussed with domain adaptation setup (Section 14.3), the max-pooling layers have been replaced with strided convolutions (as suggested in Radford et al. [50]). The stride is chosen to be 2, so that the downsampling continues to output half of convolution output dimension. Segmentation results with and without max-pooling will be presented, but max-pooling yielded terrible performance in discriminative domain adaptation, and those results are omitted when performing domain adaptation.

14.3. ADDA setup

The target map is the same architecure as the source map (in Figure 14-3). After segmentation, the parameter weights of the source map are copied into the target map before domain adaptation. The scores at that point are also reported (as 'Before DA') in the tables.

In the original u-net formulation there are max-pooling layers. However, Radford et al. [50] suggests in conditional GANs to replace any pooling layers with strided convolutions in the discriminator and fractionally-strided convolutions in the generator [50]. Even though ADDA does not have a generator in the same sense as a generative adversarial network, it definitely seems to have a very positive effect on the overall performance of the adversarial discriminative part of the model.

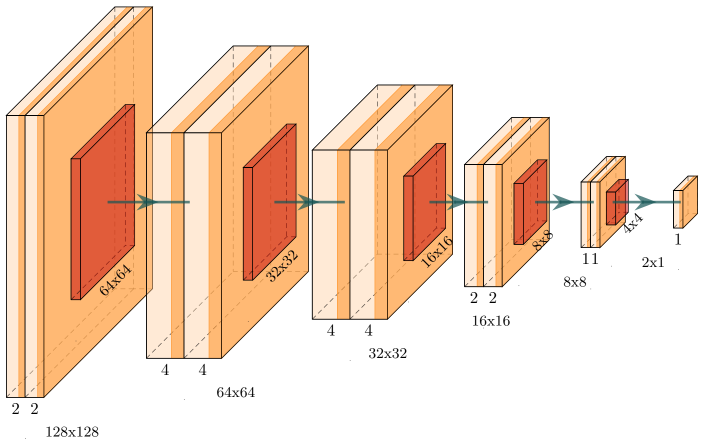


Figure 14-4: *The discriminator architecture*

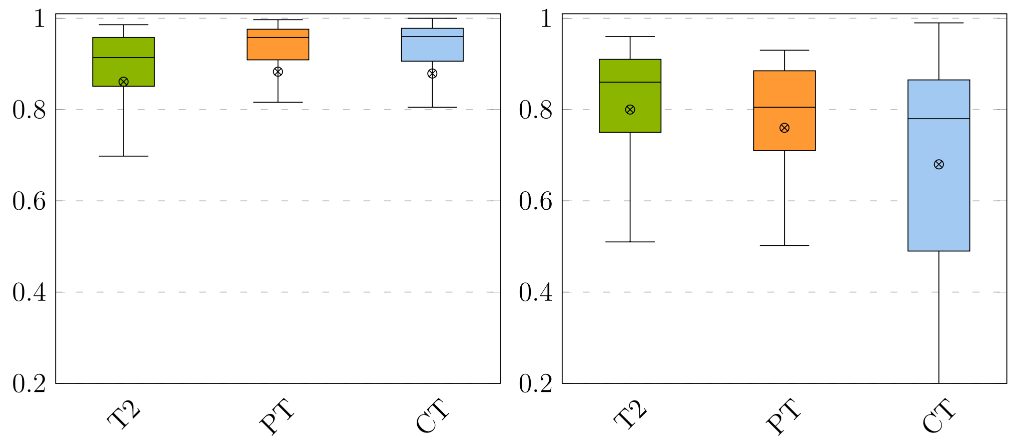
The discriminator consists of 11 convolutional layers, where the last layer is strided so that its output dimension is 2×1. The loss for the discriminator is the cross entropy loss function. However, this loss didn't seem to improve experiments performed in this thesis and has been omitted in the final implementation.

15. Results

In this section the results from performing segmentation and domain adaptation on the three modalities are presented. The segmentation results will be presented first and compared compared with the current state-of-the-art Guo et al. [2]. Domain adaptation results are presented thereafter. The discussions and conclusions regarding these results will be presented together with the results themselves. However, the discussions and conclusions directly pertaining to the proposed novelties are discussed and concluded in the final part of this thesis, Part IV.

15.1. Segmentation

Guo et al. have two papers regarding this dataset, one where they only report accuracy [93] and one where they only report the F_1 score [2].



(a) *Box plots comparing segmentation results without max pooling. For consistency, a box plot with outliers are presented in Appendix C.*

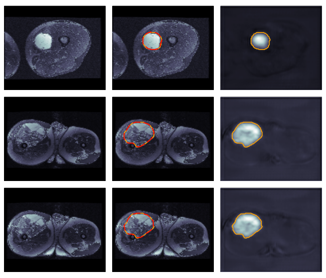
(b) *Reconstructed box plot from Guo et al. [2]. A copy of the original can be found in Appendix C.*

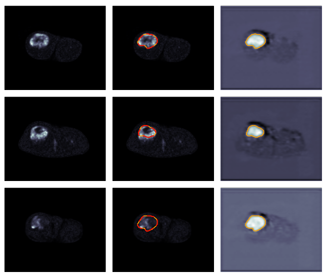
Figure 15-1: *Comparing box plots of segmentation of the image modalities. The ⊗ marks means and the horizontal lines across the boxes are medians.*

Table 15-4: *Validation data results on segmentation*

Modality	F_1			Accuracy			AUC		
	Conv	Maxpool	Guo [2]	Conv	Maxpool	Guo [93]	Conv	Maxpool	Guo
PET	**0.925**	0.924	0.760	0.982	**0.982**	0.850	0.969	**0.979**	–
T2	**0.923**	0.902	0.800	**0.981**	0.980	0.920	0.968	**0.976**	–
CT	0.844	**0.847**	0.680	0.917	**0.918**	0.770	**0.903**	0.902	–

The proposed architecture clearly performs significantly better than the current state-of-the-art in regards to F_1 score. However, it is important to note that even if Guo et al. [93] attains better accuracy in the T2 case, accuracy is a fairly unreliable measure, as the classes are heavily imbalanced. In fact, Table 15-5 suggests that their accuracy is quite worse than if the classifier were to classify all pixels as zero.


(a) *Segmentation examples of T2 image.*


(b) *Segmentation examples of PT image.*

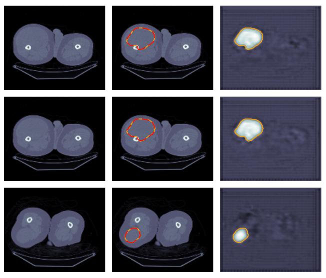

(c) *Segmentation examples of CT image.*

Figure 15-2: *Segmentation examples. Larger versions are presented in Appendix B. The orange contour is the ground truth and red is the prediction. The right hand columns show the soft assignments (probabilities) where the lighter an area is, the more probable that it contains a tumor.*

One important note however is that the withheld patients in the dataset might be different between Guo et al. [2] and the presented experiments. Furthermore, when the presented experiments were performed, the validation data and test data had the same patients. This is different from Guo et al. [2], and could potentially have an overall effect on the final results presented in the thesis. However, the experiments have been run several times and the results presented are representative for the typical output. This in turn indicates that such an effect would be rather unlikely.

Table 15-5: *Average portion of zeroes in ground truth segmentations per modality. If the classifier classified all pixels to zero, these would be the resulting accuracies. Only amongst the images actually containing a non-zero area with tumor.*

Modality	Zero-ratio
CT	0.9848
PT	0.9904
T2	0.9631

15.2. Domain adaptation

On the following pages are the tables and box plots that pertains to the results between the imaging modalities. The reader should note that there are additional figures on pages 99, 100 and 101 that contains example outputs from the domain adaptation.

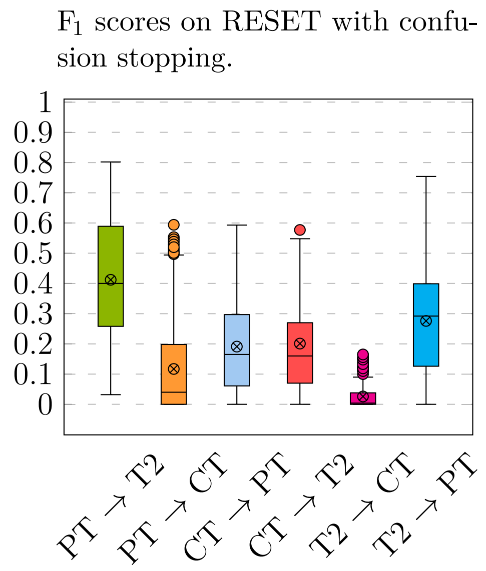
Table 15-6: *Results of modality domain adaptation. Both DA and RESET here uses the confusion stop architecture.*

Adaptation modalities	Performance metric ($\mathcal{A}$)	(15-7a) Before DA	(15-7b) DA best	(15-7c) RESET proposed
CT → PT	F_1	0.0754	0.0401	**0.2038**
	Accuracy	**0.9697**	0.9092	0.9666
	AUC	0.5116	0.5612	**0.6724**
CT → T2	F_1	0.1107	0.1407	**0.2264**
	Accuracy	**0.8829**	0.8591	0.8247
	AUC	0.4945	0.5608	**0.6422**
PT → CT	F_1	0.0292	0.0292	**0.1221**
	Accuracy	0.0149	0.0149	**0.9514**
	AUC	0.4937	0.5529	**0.5789**
PT → T2	F_1	0.3367	0.3073	**0.3922**
	Accuracy	**0.8853**	0.8849	0.8774
	AUC	0.5640	0.5738	**0.6386**
T2 → CT	F_1	**0.2024**	0.1173	0.0266
	Accuracy	**0.9368**	0.8998	0.9347
	AUC	0.7986	**0.8039**	0.7576
T2 → PT	F_1	**0.3799**	0.0268	0.2602
	Accuracy	**0.9705**	0.8504	0.9495
	AUC	0.8077	0.7570	**0.8181**

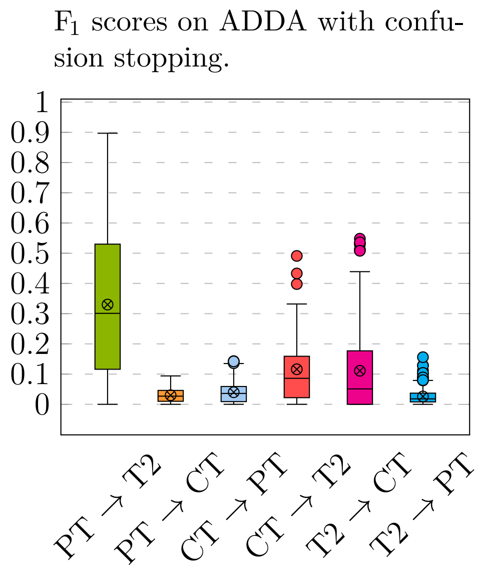
Column (15-7a):	Values of target map through trained source map before domain adaptation has begun
Column (15-7b):	The best F_1 results achieved without the optimizer parameter resets.
Column (15-7c):	ADDA compared with RESET when run with fairly similar specifications.

Table 15-7: *Overview of F_1 scores overview for domain adaptation on medical image modalities. The bold numbers show best stopping method with the current adaptation modality. Underline shows best scheme (DA / RESET) within the current stopping method. 'Highest peak' is not considered when evaluating best stopping method.*

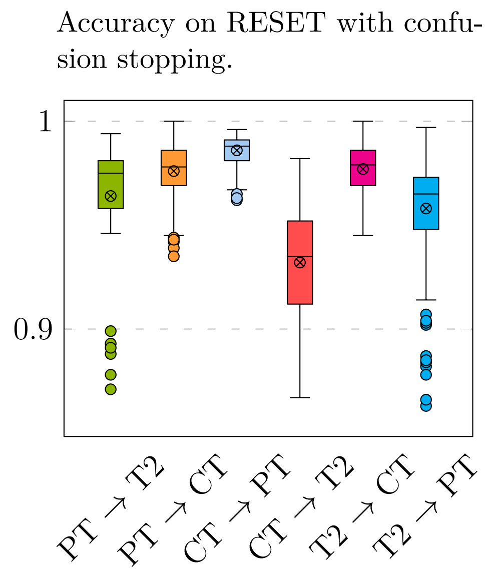
Adaptation modalities	(15-8a) Highest peak		(15-8b) Confusion stop		(15-8c) Latest stop		(15-8d) Best loss stop	
	DA	RESET	DA	RESET	DA	RESET	DA	RESET
CT → PT	0.1049	*0.3485*	0.0401	***0.2038***	0.0401	*0.2038*	*0.0186*	0.0186
CT → T2	0.2948	*0.3870*	0.1407	*0.2264*	0.0728	***0.2831***	0.0728	*0.2264*
PT → CT	*0.1656*	0.1624	0.0292	***0.1221***	0.0873	*0.0948*	0.0292	*0.1221*
PT → T2	0.2812	*0.3986*	0.3073	***0.3922***	0.2668	*0.3889*	*0.3271*	0.3089
T2 → CT	0.1919	*0.1954*	*0.1173*	0.0266	*0.0708*	0.0266	0.0292	***0.1642***
T2 → PT	0.2537	*0.5186*	0.0268	*0.2602*	0.0186	*0.2776*	0.2066	***0.4466***



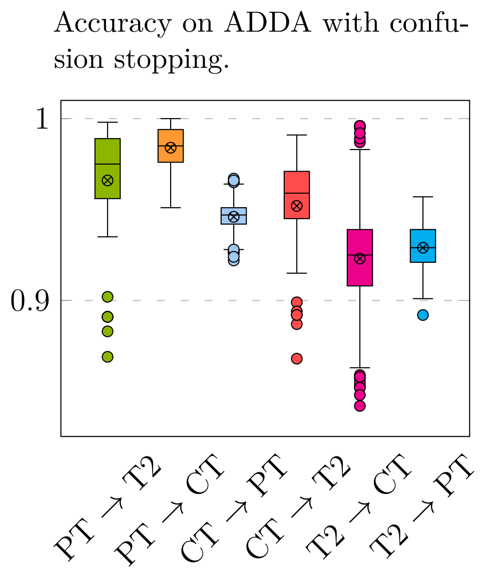
(a) *Boxplots of F_1 scores for results using the RESET scheme using confusion stopping.*



(b) *Boxplots of F_1 scores for results using the ADDA scheme using confusion stopping.*



(c) *Boxplots of accuracy for results using the RESET scheme using confusion stopping.*



(d) *Boxplots of accuracy for results using the ADDA scheme using confusion stopping.*

Figure 15-3: *Boxplots comparing scores between the ADDA and RESET schemes. The ⊗ marks means and the horizontal lines across the boxes are medians. Notice that the y-axes are very different in the bottom row. (Range of y axes different on a and b compared to c and d)*

Table 15-6 is intended to give a comparison between schemes (RESET and DA), as well as general DA results. Table 15-7, is intended to compare stopping methods.

There are of course no guarantees that any improvement will arise at all from ADDA, and considering that this is an unsupervised method applied on a small real world dataset with complex structures, the achieved results are clearly deemed satisfactory, especially in regards to the proposed improvement. As is clear in Table 15-6 is that all but AUC favor the proposed methods, which indicates that the tuning of the true positive rate and the false positive rate is better, and that for the most part it does significantly better than random guessing.

Obtaining hyper parameters for this project has been challenging. For the results to be comparable, and yet keep the project to a realistic timeframe, it seemed natural to seek an architecture configuration with accompanying hyper parameters that capture representative results from all modalities and simultaneously didn't favor one modality. Fortunately the results would consistently tend to be as presented in Table 15-7, regardless of configuration of the U-net. However, if one were to apply different architectures, or even just changed the setups, the individual results might improve even more.

To the author's best knowledge, performing unsupervised domain adaptation on this dataset has not been done before.

One interesting observation in Table 15-6 is that the classification accuracy tends to decline after domain adaptation. After domain adaptation, there are more positive pixels within the regions of where the tumor is (see table Table 15-5). This might indicate that the discriminator is learning on the underlying structure of the output, which is what we desire, and not on discriminating on the mere number of correctly classified pixels. This is further reinforced with the increasing in F_1 score and the fact that the decrease is relatively small. Inspection of the result figures (see on pages 99 and 100 respectively), further supports this line of reasoning.

The F_1 score will tend to reward true positives more than it will punish false positives, as by design, it will be strict with smaller areas. Figure 15-6 actually demonstrates this: when there are no pixels classified overlapping the ground truth tumor, the F_1 score is equal (or very close to) zero. Additionally, it is easy to argue that in tumor classification and segmentation, it is preferable to present too many false positives over false negatives.

A related observation in Table 15-6 is that on T2 $\rightarrow$ CT the classification accuracy increases whilst the F_1 score decreased after domain adaptation. As indicated by Table 15-5, this accuracy is lower than if the entire image classified to 0. This might suggest that larger parts of the image were false positives before domain adaptation. In that state, the tumor might have been trivially covered (if big areas of false positives were present). However after domain adaptation, the number of positive classified pixels could have been reduced, but this at the expense of removing true positive classified parts of the tumor. In turn one might take this to indicate that it has not been training sufficiently long, but Table 15-7 suggests that it might not be the case because the the best loss stopping criterion yielded significantly better results, however it is still decreased from the initial 'before DA' value.

A possible explanation is that the discriminator didn't really get properly confused, and that the loss only kept increasing after the initial training. This way the training algorithm would have obtain the best loss value at an early epoch. Upon further inspection of the RESET results, at the best loss, the target map had only iterated 1914 iterations (7 epochs), whilst the target map confusion was stopped at 8486 iterations (33 epochs) out of a total of 12903 iterations (51 epochs), which clearly support this hypothesis.

This is an indication of suboptimal hyper parameters. And as this result is similar to the case for the other adaptation where T2 is the source dataset, this indicates that the results might be improved by modifying these specific architecture and hyper parameters.

This could in fact be caused by the target map becoming too dependent on momentum, and goes in suboptimal directions. The interpretation is that when training the target map via the discriminator, it might focus excessively on a too high-level part the underlying structure. For instance, the discriminator might recognize data only corresponding (when classified) to coherent 'blobs', but the location of the blobs aren't impactful enough during training. The images in Figure 15-5 are also consistent with these conclusions. In such an event it would would likely cause the RESET to perform worse as it relies less heavily on momentum terms, and when comparing Figure 15-3d with Figure 15-3c, it is consistent with what you would expect in this case as well. Tuning momentum term parameters β_1 and β_2 in (7) and (8) respectively and/or the momentum parameter resetting period, might improve this result as well.

Although the results pertaining to ADDA are novel and as such could be considered state-of-the-art, some of the results might not prove too challenging to improve by tuning hyper parameters. Hopefully this thesis provide a solid foundation to improve on said results, which is clearly encouraged by the author.

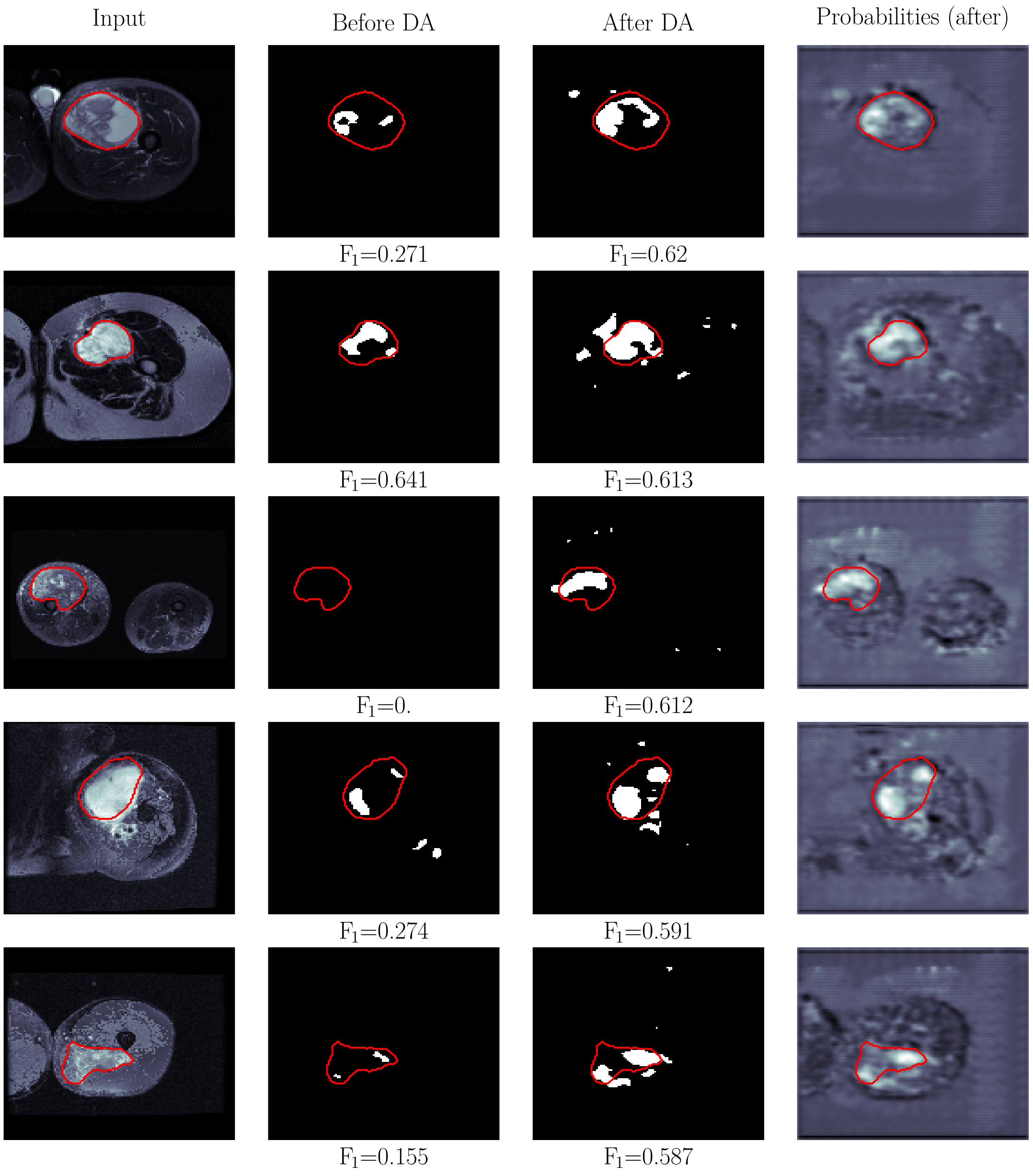


Figure 15-4: *PT → T2 domain adaptation results with randomly selected images with F_1 scores between the 60th and 80th percentile. Ground truth is marked with the red line.*

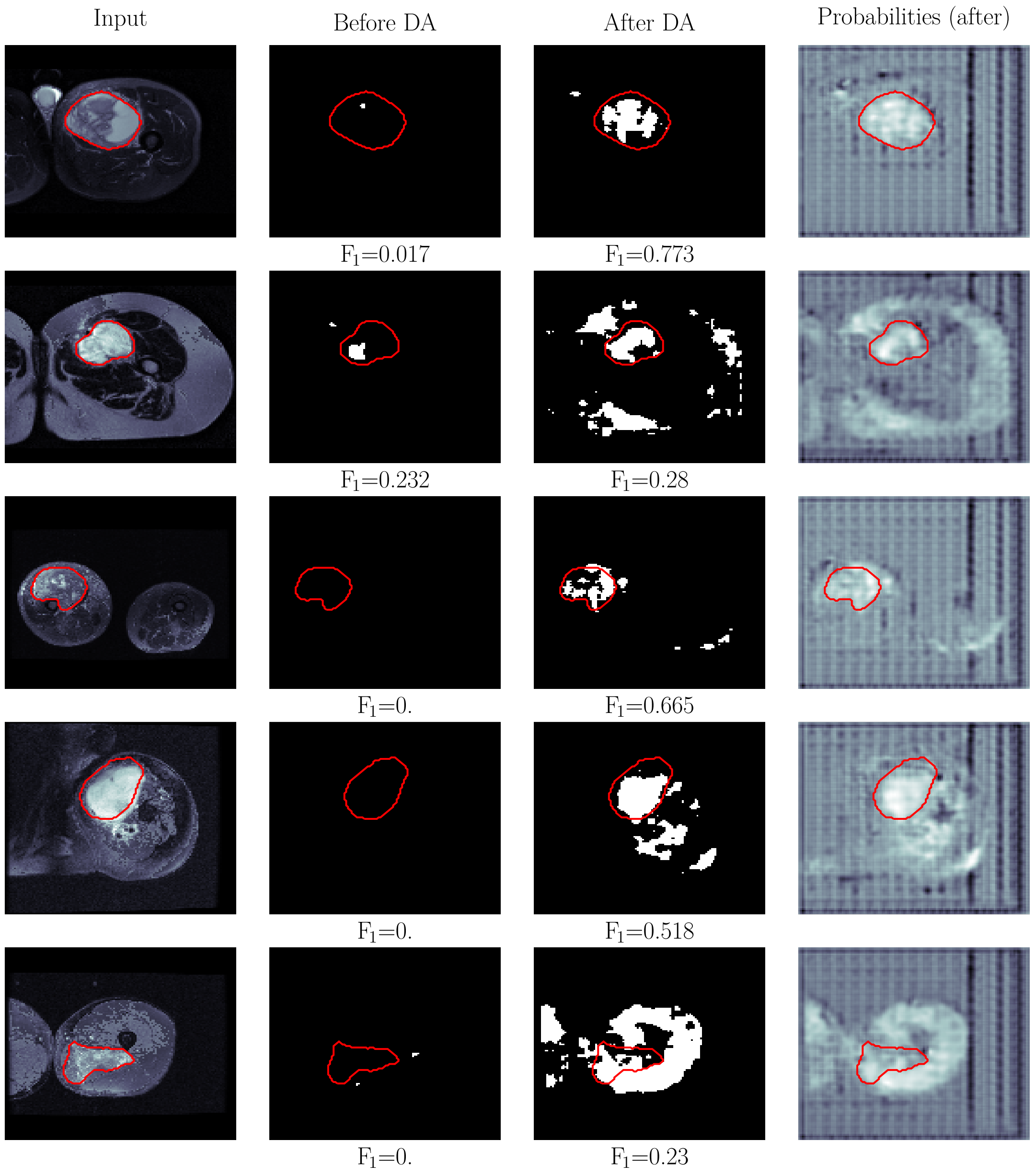


Figure 15-5: *CT → T2 domain adaptation results with randomly selected images with F_1 scores between the 60th and 80th percentile. Ground truth is marked with the red line.*

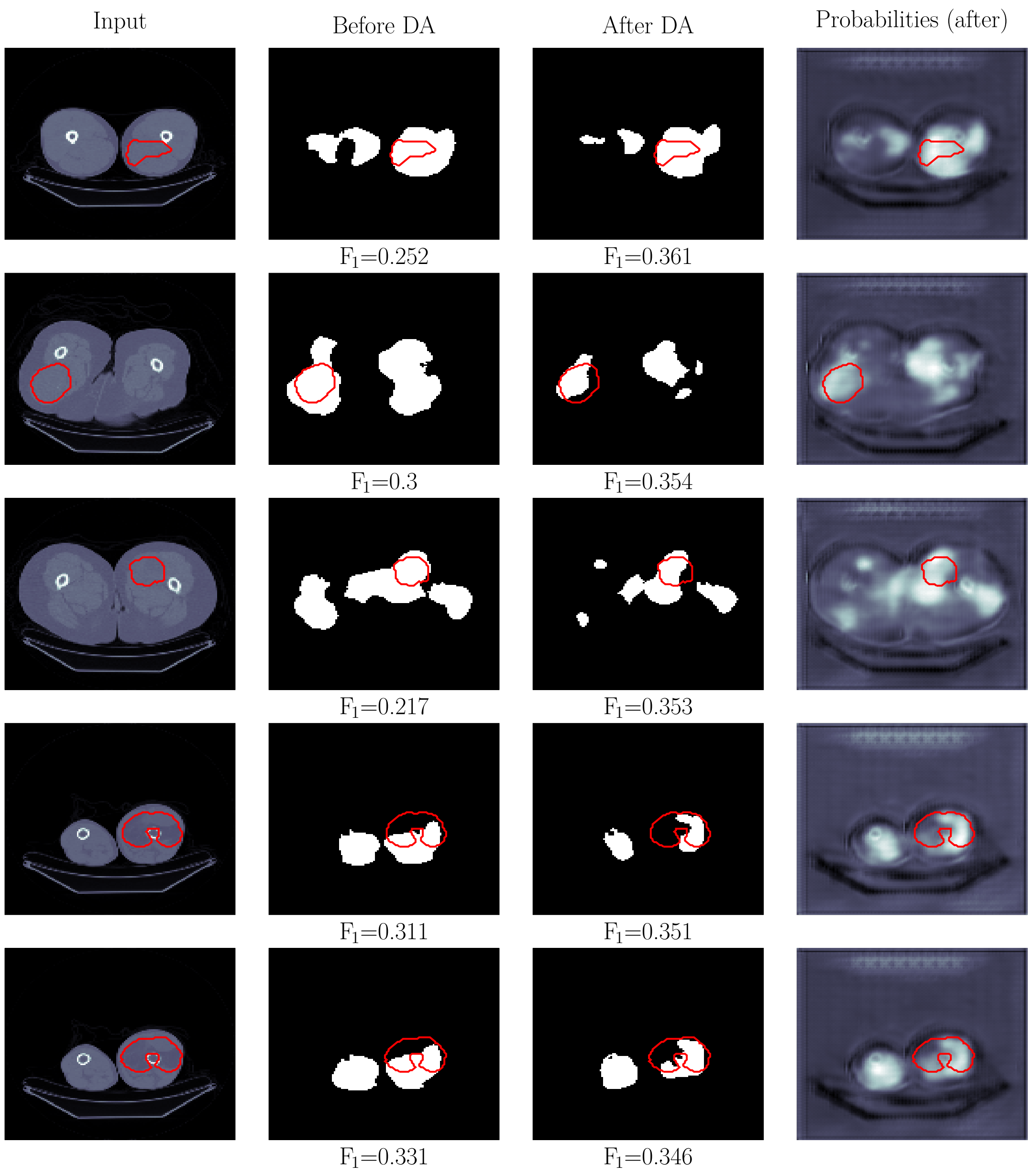


Figure 15-6: *T2 $\rightarrow$ CT domain adaptation results with randomly selected images with F_1 scores between the 60th and 80th percentile. Ground truth is marked with the red line.*

Part V / Discussion and concluding remarks

In this final part, the proposed hypotheses presented in Section 3 will be discussed in light of the results presented in Part IV. The performance of the novelties will first be discussed individually and finally, a conclusion of the novelties and the thesis as a whole.

16. Momentum reset discussion

As demonstrated throughout Part IV, the momentum reset scheme did tend to perform better in the experiments shown above.

However, the resetting of momentum parameters was intended as a demonstration of the problem, and that is has potential to be improved upon. In this regard, there are countless other approaches in regards to improving momentum in adversarial discriminative training. One could, for example, do an upper capping on the values, some variant of simulated annealing or design custom optimizers.

Another interesting approach would be to introduce an optimizer that has a short time memory. For instance, adjusting the Adagrad [94] optimizer to only account for the last n steps. The Adagrad update rule is

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \frac{\eta \odot \frac{\partial \mathcal{E}}{\partial \boldsymbol{\theta}^{(t)}}}{\sqrt{\mathbf{G}^{(t)}}}, \tag{27}$$

with $\mathbf{G}_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix with each diagonal element is the sum of the squares of the gradients up to time step t: $G_{i,i}^{(t)} = \sum_{\tau=0}^{t} \frac{\partial \mathcal{E}}{\partial \boldsymbol{\theta}^{(\tau)}}$. In (27), both the division and square root is element wise with $\mathbf{G}^{(t)}$.

Adagrad's main weakness is that the accumulation of the squared gradients in $\mathbf{G}^{(t)}$ becomes very big and thus the learning rate will shrink and eventually disappear, as Ruder [58] points out. It is assumed that limiting the lower time step has been attempted, and might not have given satisfactory results, but might do so in the case of momentum in adversarial discriminative training.

As shown in the experiments, the momentum reset has some flaws. Firstly it does not perform well if the architectures relies on momentum to get good results, as shown in the experiments. Secondly, it might have issues getting stuck in saddle points if the moment parameters suddenly are reset near such a point. If there is a 'rough patch' in the loss function, or the loss function is of a rough nature, resetting the momentum term could have a very negative effect. As earlier stated, addressing this momentum weakness in adversarial discriminative, does not mean that momentum should be taken out of the optimizer equation, but rather that it should be handled as a separate problem.

17. Confusion score discussion

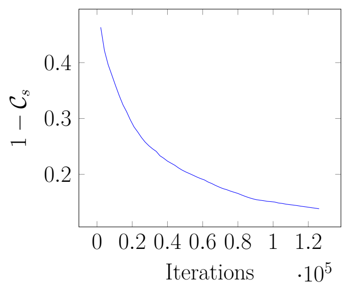
For convenience, the expression for confusion score is repeated:

$$\mathcal{C}_s^* = \mathbb{E}\left[1 - \left|\frac{t_p}{p} - \frac{1}{2}\right| - \left|\frac{f_p}{n} - \frac{1}{2}\right|\right]. \tag{25}$$

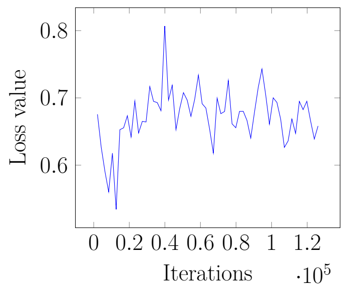
The confusion measure demonstrated a clear tendency to follow a stable, monotonic structure compared to the discriminator loss and accuracy, that were clearly more unpredictable. In Figure 17-1 it shows typical examples of how the loss and accuracy could remain unpredictable, whilst the confusion score remains stable.

It is important to note that the loss here is from the discriminator and would likely be inversely correlated with the target map loss.

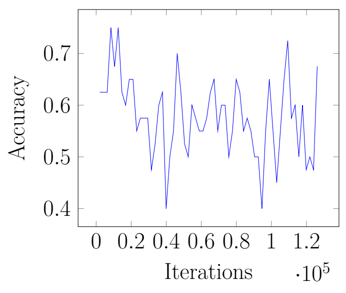
Determining stopping criterion for unsupervised functions are hard. Even though confusion score has been well behaved in experiments presented in this thesis, it would definitely not mean that it is the optimal solution to the convergence problem of training discriminative adversarial models. It has however proven useful as a smooth measurement that reports the state of the discriminator quite well in ADDA on the soft tissue sarcoma dataset.



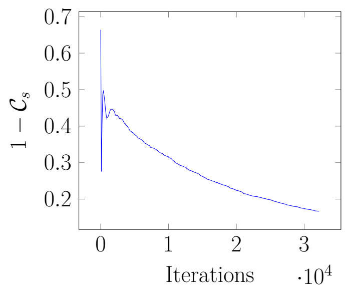
(a) *Confusion score $(1 - \mathcal{C}_s)$ from one of the experiments.*



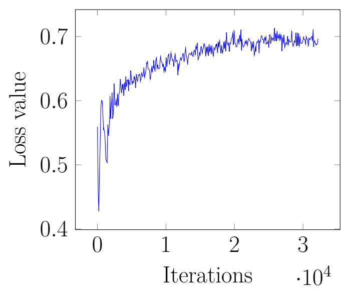
(b) *Loss from one of the experiments.*



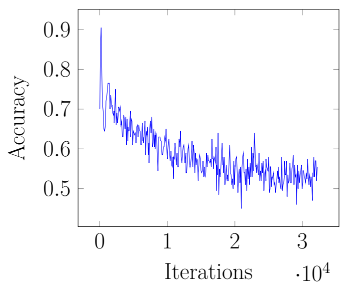
(c) *Accuracy from one of the experiments.*



(d) *Confusion score $(1 - \mathcal{C}_s)$ from one of the experiments.*



(e) *Loss from one of the experiments.*



(f) *Accuracy from one of the experiments.*

Figure 17-1: *Values of confusion score and loss against training iterations for two different experiments using the RESET scheme. The same rows have the same experiments. All from the discriminators, sampled at regular intervals.*

18. Conclusion

The momentum reset did improve on the state-of-the-art technique ADDA. However, there might be other approaches to momentum reset which produces superior results. The momentum reset scheme presented in this thesis is hopefully regarded as solid preliminary work for such methods.

The confusion score demonstrated powerful performance in these experiments, and being fairly lightweight, it might prove useful for training discriminators in the future. Of course, obtaining a measurement that in a general way provides the optimal solution to adversarial discriminative training might prove troublesome, if not impossible, and the confusion is not intended to solve that problem. However, it might be a step in the right direction in regards to approaching training termination, and demonstrates that perhaps the loss isn't necessarily what should be the main focus when obtaining the stopping criterion.

This thesis set out to perform segmentation, unsupervised adversarial discriminative domain adaptation, as well as set the groundwork fro improving upon the state-of-the-art techniques for doing adversarial discriminative training. The segmentation, to the author's best knowledge, achieved state-of-the-art results. The ADDA architecture had a strong tendency to improve the performance on the classification after unsupervised training.

Furthermore, the confusion score demonstrated a trend of being more reliable as a stopping criterion for discriminative training, than modern state-of-the-art techniques.

The momentum reset scheme clearly demonstrates and address an important flaw in state-of-the-art techniques for doing adversarial discriminative training. Additionally it tended to consistently provide good results on the medical image modalities, but performed inferior to state of the art methods on the simpler dataset, arguably due to the model relying heavily on momentum on these architectures.

This concludes this master's thesis.

/ Appendices

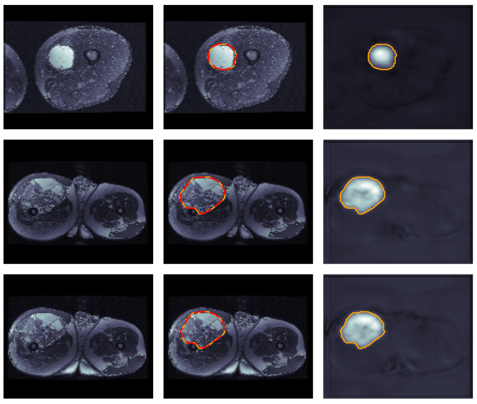
Appendix A. Source code

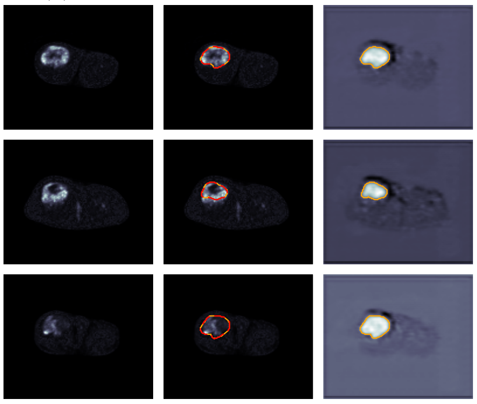
For the sake of transparency, and reproducibility, the code is uploaded to github: `Strauman/Masters-thesis` [i].

The final codebase accompanying this thesis contains over 15 000 lines of code. With a lot of moving parts when comparing three times three modalities, two schemes and three different stopping criteria, it might prove challenging to navigate. The author is happy to provide answers to any questions or other inquiries in regards to the code or any other parts of this work.

[i] `https://github.com/Strauman/Masters-thesis`

Appendix B. Segmentation plots


(a) *Segmentation examples of T2 image.*


(b) *Segmentation examples of PT image.*

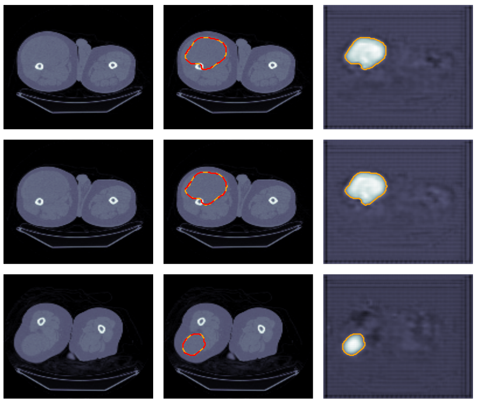

(c) *Segmentation examples of CT image.*

Figure B-1: *The orange contour is the ground truth and red is the prediction. The right hand columns show the soft assignments (probabilities) where the lighter an area is, the more probable that it contains a tumor.*

Appendix C. Box plot from Guo et al. [2]

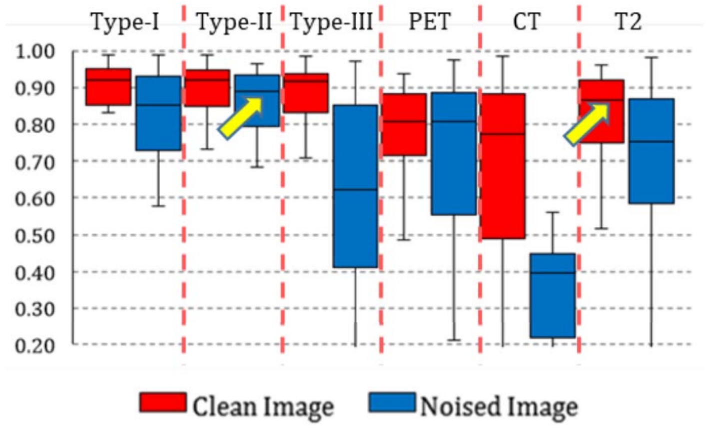


Figure C-1: *The original boxplot that Figure Appendix C is based upon. The yellow arrows are a part of the original figure. Original box plot (Fig. 6.) from Deep Learning-Based Image Segmentation on Multimodal Medical Imaging by Guo et al. [2]. The red boxes in the right half of the figure (with title PET, CT and T2) are DICE segmentation result.*

References

[1] Kenji Suzuki. Overview of deep learning in medical imaging. *Radiological Physics and Technology*, 10(3):257–273, September 2017. ISSN 1865-0341. doi: `10.1007/s12194-017-0406-5`.

[2] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li. Deep Learning-Based Image Segmentation on Multimodal Medical Imaging. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 3(2):162–169, March 2019. ISSN 2469-7311. doi: `10.1109/TRPMS.2018.2890359`.

[3] MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. `http://yann.lecun.com/exdb/mnist/`, .

[4] The Street View House Numbers (SVHN) Dataset. `http://ufldl.stanford.edu/housenumbers/`, .

[5] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994. ISSN 0162-8828. doi: `10.1109/34.291440`.

[6] Ashish K. Jha, Catherine M. DesRoches, Peter D. Kralovec, and Maulik S. Joshi. A Progress Report On Electronic Health Records In U.S. Hospitals. *Health Affairs*, 29(10):1951–1957, October 2010. ISSN 0278-2715. doi: `10.1377/hlthaff.2010.0502`.

[7] Craig A. Pedersen, Philip J. Schneider, and Douglas J. Scheckelhoff. ASHP national survey of pharmacy practice in hospital settings: Prescribing and transcribing—2016. *American Journal of Health-System Pharmacy*, page ajhp170228, July 2017. ISSN 1079-2082, 1535-2900. doi: `10.2146/ajhp170228`.

[8] Tim Dall, Terry West, Ritashree Chakrabarti, Ryan Reynolds, and Will Lacobucci. 2018 Update: The Complexities of Physician Supply and Demand: Projections from 2016 to 2030. *Association of American Medical Colleges*, page 66, March 2018. Prepared and submitted by IHS Markit Ltd.

[9] Stephen M. Petterson, Winston R. Liaw, Robert L. Phillips, David L. Rabin, David S. Meyers, and Andrew W. Bazemore. Projecting US primary care physician workforce needs: 2010-2025. *Annals of Family Medicine*, 10(6): 503–509, 2012 Nov-Dec. ISSN 1544-1717. doi: `10.1370/afm.1431`.

[10] Jerome A. Osheroff, Jonathan M. Teich, Blackford Middleton, Elaine B. Steen, Adam Wright, and Don E. Detmer. A roadmap for national action on clinical decision support. *Journal of the American Medical Informatics Association: JAMIA*, 14(2):141–145, 2007 Mar-Apr. ISSN 1067-5027. doi: `10.1197/jamia.M2334`.

[11] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, December 2017. ISSN 1361-8415. doi: `10.1016/j.media.2017.07.005`.

[12] Miles Wernick, Yongyi Yang, Jovan Brankov, Grigori Yourganov, and Stephen Strother. Machine Learning in Medical Imaging. *IEEE Signal Processing Magazine*, 27(4):25–38, July 2010. ISSN 1053-5888. doi: `10.1109/MSP.2010.936730`.

[13] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, February 2017. ISSN 1476-4687. doi: `10.1038/nature21056`.

[14] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv:1711.05225 [cs, stat]*, November 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015,*

Santiago, Chile, December 7-13, 2015, pages 1026–1034. IEEE Computer Society, 2015. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123.

[16] Robert D. Hof. Is Artificial Intelligence Finally Coming into Its Own? https://www.technologyreview.com/s/513696/deep-learning/. Accessed at 2018-11-24 13:21:00.

[17] H. Greenspan, B. van Ginneken, and R. M. Summers. Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, May 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2553401.

[18] Mark A. Musen, Blackford Middleton, and Robert A. Greenes. Clinical Decision-Support Systems. In Edward H. Shortliffe and James J. Cimino, editors, *Biomedical Informatics: Computer Applications in Health Care and Biomedicine*, pages 643–674. Springer London, London, 2014. ISBN 978-1-4471-4474-8. doi: 10.1007/978-1-4471-4474-8_22.

[19] Andreas Storvik Strauman, Filippo Maria Bianchi, Karl Øyvind Mikalsen, Michael Kampffmeyer, Cristina Soguero-Rúız, and Robert Jenssen. Classification of postoperative surgical site infections from blood measurements with missing data using recurrent neural networks. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics, BHI 2018, Las Vegas, NV, USA, March 4-7, 2018*, pages 307–310. IEEE, 2018. ISBN 978-1-5386-2405-0. doi: 10.1109/BHI.2018.8333430.

[20] S.-B. Lo, S.-A. Lou, Jyh-Shyan Lin, M. T. Freedman, M. V. Chien, and S. K. Mun. Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, 14(4): 711–718, December 1995. ISSN 0278-0062. doi: 10.1109/42.476112.

[21] *Concise Medical Dictionary*. Oxford University Press, 8 edition, January 2010. ISBN 978-0-19-955714-1. doi: 10.1093/acref/9780199557141.001.0001. Obtained via ordnett.no.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C.

Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836, 1476-4687. doi: `10.1038/nature14539`.

[24] Hongkai Wang, Zongwei Zhou, Yingci Li, Zhonghua Chen, Peiou Lu, Wenzhi Wang, Wanyu Liu, and Lijuan Yu. Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT images. *EJNMMI Research*, 7(1):11, January 2017. ISSN 2191-219X. doi: `10.1186/s13550-017-0260-9`.

[25] Wei Yang, Yingyin Chen, Yunbi Liu, Liming Zhong, Genggeng Qin, Zhentai Lu, Qianjin Feng, and Wufan Chen. Cascade of multi-scale convolutional neural networks for bone suppression of chest radiographs in gradient domain. *Medical Image Analysis*, 35:421–433, January 2017. ISSN 1361-8415. doi: `10.1016/j.media.2016.08.004`.

[26] Ozan Oktay, Wenjia Bai, Matthew Lee, Ricardo Guerrero, Konstantinos Kamnitsas, Jose Caballero, Antonio de Marvao, Stuart Cook, Declan O'Regan, and Daniel Rueckert. Multi-input Cardiac Image Super-Resolution Using Convolutional Neural Networks. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Lecture Notes in Computer Science, pages 246–254. Springer International Publishing, 2016. ISBN 978-3-319-46726-9.

[27] Jian Ren, Ilker Hacihaliloglu, Eric A. Singer, David J. Foran, and Xin Qi. Adversarial Domain Adaptation for Classification of Prostate Histopathology Whole-Slide Images. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*,

Lecture Notes in Computer Science, pages 201–209. Springer International Publishing, 2018. ISBN 978-3-030-00934-2.

[28] Yue Zhang, Shun Miao, Tommaso Mansi, and Rui Liao. Task Driven Generative Modeling for Unsupervised Domain Adaptation: Application to X-ray Image Segmentation. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 599–607. Springer International Publishing, 2018. ISBN 978-3-030-00934-2.

[29] Bingsheng Huang, Martin Wai-Ming Law, and Pek-Lan Khong. Whole-Body PET/CT Scanning: Estimation of Radiation Dose and Cancer Risk. *Radiology*, 251(1):166–174, April 2009. ISSN 0033-8419. doi: `10.1148/radiol.2511081300`.

[30] Rongjian Li, Wenlu Zhang, Heung-Il Suk, Li Wang, Jiang Li, Dinggang Shen, and Shuiwang Ji. Deep Learning Based Imaging Data Completion for Improved Brain Disease Diagnosis. In Polina Golland, Nobuhiko Hata, Christian Barillot, Joachim Hornegger, and Robert Howe, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, Lecture Notes in Computer Science, pages 305–312. Springer International Publishing, 2014. ISBN 978-3-319-10443-0.

[31] Dong Nie, Xiaohuan Cao, Yaozong Gao, Li Wang, and Dinggang Shen. Estimating CT Image from MRI Data Using 3D Fully Convolutional Networks. In Gustavo Carneiro, Diana Mateus, Loïc Peter, Andrew Bradley, João Manuel R. S. Tavares, Vasileios Belagiannis, João Paulo Papa, Jacinto C. Nascimento, Marco Loog, Zhi Lu, Jaime S. Cardoso, and Julien Cornebise, editors, *Deep Learning and Data Labeling for Medical Applications*, volume 10008, pages 170–178. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46975-1 978-3-319-46976-8. doi: `10.1007/978-3-319-46976-8_18`.

[32] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Se-

mantic Segmentation using Adversarial Networks. *arXiv:1611.08408 [cs]*, November 2016.

[33] Nanqing Dong, Michael Kampffmeyer, Xiaodan Liang, Zeya Wang, Wei Dai, and Eric Xing. Unsupervised Domain Adaptation for Automatic Estimation of Cardiothoracic Ratio. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 544–552. Springer International Publishing, 2018. ISBN 978-3-030-00934-2.

[34] Dwarikanath Mahapatra, Behzad Bozorgtabar, Jean-Philippe Thiran, and Mauricio Reyes. Efficient Active Learning for Image Classification and Segmentation Using a Sample Selection and Conditional Generative Adversarial Network. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 580–588. Springer International Publishing, 2018. ISBN 978-3-030-00934-2.

[35] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer Science & Business Media, December 2012. ISBN 978-1-4612-0745-0.

[36] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken. Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks. *IEEE Transactions on Medical Imaging*, 35(5):1160–1169, May 2016. ISSN 0278-0062. doi: `10.1109/TMI.2016.2536809`.

[37] Wei Shen, Mu Zhou, Feng Yang, Caiyun Yang, and Jie Tian. Multi-scale Convolutional Neural Networks for Lung Nodule Classification. In Sebastien Ourselin, Daniel C. Alexander, Carl-Fredrik Westin, and M. Jorge Cardoso, editors, *Information Processing in Medical Imaging*, volume 9123, pages 588–

599. Springer International Publishing, Cham, 2015. ISBN 978-3-319-19991-7 978-3-319-19992-4. doi: 10.1007/978-3-319-19992-4_46.

[38] Dong Nie, Han Zhang, Ehsan Adeli, Luyan Liu, and Dinggang Shen. 3D Deep Learning for Multi-modal Imaging-Guided Survival Time Prediction of Brain Tumor Patients. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Lecture Notes in Computer Science, pages 212–220. Springer International Publishing, 2016. ISBN 978-3-319-46723-8.

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*, November 2014.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241. Springer International Publishing, 2015. ISBN 978-3-319-24574-4.

[41] Tom Brosch, Jochen Peters, Alexandra Groth, Thomas Stehle, and Jürgen Weese. Deep Learning-Based Boundary Detection for Model-Based Segmentation with Application to MR Prostate Segmentation. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 515–522. Springer International Publishing, 2018. ISBN 978-3-030-00937-3.

[42] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, Lecture Notes in Computer

Science, pages 424–432. Springer International Publishing, 2016. ISBN 978-3-319-46723-8.

[43] Puyang Wang, Vishal M. Patel, and Ilker Hacihaliloglu. Simultaneous Segmentation and Classification of Bone Surfaces from Ultrasound Using a Multi-feature Guided CNN. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 134–142. Springer International Publishing, 2018. ISBN 978-3-030-00937-3.

[44] E. Hosseini-Asl, R. Keynton, and A. El-Baz. Alzheimer's disease diagnostics by adaptation of 3D convolutional network. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 126–130, September 2016. doi: `10.1109/ICIP.2016.7532332`.

[45] Harry Pratt, Frans Coenen, Deborah M. Broadbent, Simon P. Harding, and Yalin Zheng. Convolutional Neural Networks for Diabetic Retinopathy. *Procedia Computer Science*, 90:200–205, January 2016. ISSN 1877-0509. doi: `10.1016/j.procs.2016.07.014`.

[46] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. *arXiv:1702.05464 [cs]*, February 2017.

[47] Vivek Kumar Singh, Santiago Romani, Hatem A. Rashwan, Farhan Akram, Nidhi Pandey, Md. Mostafa Kamal Sarker, Saddam Abdulwahab, Jordina Torrents-Barrena, Adel Saleh, Miguel Arquez, Meritxell Arenas, and Domenec Puig. Conditional Generative Adversarial and Convolutional Networks for X-ray Breast Mass Segmentation and Shape Classification. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, Lecture Notes in Computer Science, pages 833–840. Springer International Publishing, 2018. ISBN 978-3-030-00934-2.

[48] Aaron Chadha and Yiannis Andreopoulos. Improving Adversarial Discriminative Domain Adaptation. *arXiv:1809.03625 [cs]*, September 2018.

[49] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, November 2015.

[51] Google scholar search for unsupervised representation learning radford. `https://scholar.google.no/scholar?hl=en&as_sdt=0%2C5&as_vis=1&q=unsupervised+representation+learning+radford`, July 2019.

[52] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014.

[53] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[55] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Advanced Series. Brooks/Cole, Cengage Learning, Belmont, Calif., 2. ed., internat. student ed., [nachdr.] edition, 20. ISBN 978-0-495-39187-6. OCLC: 935033840.

[56] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the Loss Landscape of Neural Nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors,

Advances in Neural Information Processing Systems 31, pages 6389–6399. Curran Associates, Inc., 2018.

[57] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[58] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, September 2016.

[59] R. SUTTON. Two problems with back propagation and other steepest descent learning procedures for networks. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 823–832, 1986.

[60] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, January 1999. ISSN 0893-6080. doi: `10.1016/S0893-6080(98)00116-6`.

[61] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. ON THE CONVERGENCE OF ADAM AND BEYOND. page 23, 2018.

[62] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised Learning. In *The Elements of Statistical Learning*, pages 1–101. Springer New York, New York, NY, 2009. ISBN 978-0-387-84857-0 978-0-387-84858-7. doi: `10.1007/b94608_14`.

[63] Ulrike von Luxburg, Robert C. Williamson, and Isabelle Guyon. Clustering: Science or Art? In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 65–79, June 2012.

[64] Daniel J. Trosten, Andreas Storvik Strauman, Michael Kampffmeyer, and Robert Jenssen. Recurrent Deep Divergence-based Clustering for Simultaneous Feature Learning and Clustering of Variable Length Time Series. In *IEEE International Conference on Acoustics, Speech and Signal Processing,*

ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019, pages 3257–3261. IEEE, 2019. ISBN 978-1-4799-8131-1. doi: `10.1109/ICASSP.2019.8682365`.

[65] M. Kampffmeyer, S. Løkse, F. M. Bianchi, L. Livi, A. Salberg, and R. Jenssen. Deep divergence-based clustering. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, September 2017. doi: `10.1109/MLSP.2017.8168158`.

[66] Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, third edition edition, 2014. ISBN 978-0-262-02818-9.

[67] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[68] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Elsevier Acad. Press, Amsterdam, 4. ed edition, 2009. ISBN 978-1-59749-272-0. OCLC: 550588366.

[69] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv:1603.07285 [cs, stat]*, March 2016.

[70] Improving predictive inference under covariate shift by weighting the log-likelihood function - ScienceDirect. `https://www.sciencedirect.com/science/article/pii/S0378375800001154`, .

[71] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[72] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN 978-0-387-84857-0.

[73] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[74] An Open Source Machine Learning Framework for Everyone: Tensorflow/tensorflow. `tensorflow`, July 2019.

[75] Geof H. Givens and Jennifer A. Hoeting. *Computational Statistics*. Wiley Series in Computational Statistics. Wiley, Hoboken, N.J, 2nd ed edition, 2013. ISBN 978-0-470-53331-4.

[76] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[77] Siddharth Narayanaswamy, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank D. Wood, and Philip H. S. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5927–5937, 2017.

[78] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016.

[79] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional Image Generation with PixelCNN Decoders. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg,

Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016.

[80] Ćıcero Nogueira dos Santos, Youssef Mroueh, Inkit Padhi, and Pierre L. Dognin. Learning Implicit Generative Models by Matching Perceptual Features. *CoRR*, abs/1904.02762, 2019.

[81] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic Segmentation using Adversarial Networks. *CoRR*, abs/1611.08408, 2016.

[82] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1994–2003. PMLR, 2018.

[83] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1139–1147. JMLR.org, 2013.

[84] Martin Vallières, Carolyn R. Freeman, Sonia R. Skamene, and Issam El Naqa. A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities, 2015.

[85] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The Cancer Imaging Archive

(TCIA): Maintaining and Operating a Public Information Repository. *Journal of Digital Imaging*, 26(6):1045–1057, December 2013. ISSN 0897-1889, 1618-727X. doi: `10.1007/s10278-013-9622-7`.

[86] All CIODs – DICOM Standard Browser. `https://dicom.innolitics.com/ciods`, .

[87] Using TFRecords and tf.Example | TensorFlow Core | TensorFlow. `https://www.tensorflow.org/tutorials/load_data/tf_records`, .

[88] Python Release Python 3.6.8. `https://www.python.org/downloads/release/python-368/`, .

[89] An Open Source Machine Learning Framework for Everyone: Tensorflow/tensorflow. `tensorflow`, July 2019.

[90] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[91] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 0018-9219. doi: `10.1109/5.726791`.

[92] Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). *arXiv:1803.08375 [cs, stat]*, March 2018.

[93] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li. Medical image segmentation based on multi-modal convolutional neural network: Study on image fusion schemes. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 903–907, April 2018. doi: `10.1109/ISBI.2018.8363717`.

[94] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In Adam Tauman

Kalai and Mehryar Mohri, editors, *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 257–269. Omnipress, 2010. ISBN 978-0-9822529-2-5.

[95] M Vallières, C R Freeman, S R Skamene, and I El Naqa. A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities. *Physics in Medicine and Biology*, 60(14):5471–5496, July 2015. ISSN 0031-9155, 1361-6560. doi: 10.1088/0031-9155/60/14/5471.