Imperial College London

BioMedIA

Reliable Machine Learning for Medical Imaging Data through Automated Quality Control and Data Harmonization.

Author Robert Robinson
Submitted May 22nd 2020

Supervisors Dr. Ben Glocker[1]
Dr. Chris Page[2]
Dr. Marius de Groot[2]
Prof. Daniel Rueckert[1]

[1]BioMedIA, Department of Computing, Imperial College London, UK.
[2]GlaxoSmithKline Research and Development, Uxbridge, UK.

A thesis submitted in fulfilment of the requirements for the degree of *Philosophiae Doctor*

Biomedical Image Analysis Group, Department of Computing,
Imperial College London.

BioMedIA,
Department of Computing,
Huxley Building,
Imperial College London,
180 Queens Gate,
South Kensington,
SW7 2AZ

A learning experience is one of those things that says,
'You know that thing you just did? Don't do that.'

Douglas Adams - *The Salmon of Doubt*

Many learning experiences occurred conducting this research.

For all of their valuable support, this thesis, the fruit of these last 5 years, is dedicated to my partner, Nick, my parents, Margie and Eric, and my friends. It is also dedicated to Danny, my constant companion.

I would never have lived to begin, much less finish, the work herein if it weren't for my Dad; he is a true life-saver and superhero.

Declaration

The works presented in this thesis are original ideas fostered jointly between the author and their supervisors. To the best of the author's knowledge, the work is original and attribution has been given where appropriate. Where others have contributed to the work, it has been acknowledged in co-authorship of the relevant papers outlined here.

The work in chapter 3 was published in the conference proceedings of the 20th international conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) in 2017. The work in chapter 4 was published in the conference proceedings of the 21st international conference on MICCAI in 2018. The work from chapter 3 was extended incorporating work from chapter 4 into the article published in the Journal of Cardiovascular Magnetic Resonance (JCMR) in 2019. The work in chapter 5 is submitted to the 23rd international conference on MICCAI in 2020. The work in this thesis has been presented as an abstract in poster and oral form at various conferences and events.

This document was written solely by the author unless otherwise acknowledged. All figures and tables have been generated by the author. The document has been formatted and styled by the author.

Some illustrations, generated by the author, have been previously uploaded to their personal website at https://mlnotebook.github.io under a Creative Commons Attribution-Non Commercial 4.0 International Licence.

This thesis has been submitted in fulfilment of the requirements for the degree of *Philosophiae Doctor* at Imperial College London only. It has not been submitted for any other degree or professional qualification.

Copyright

Acknowledgements

The contributions of this thesis were funded by the Engineering and Physical Science Research Council (EPSRC) Centre for Doctoral Training (CDT) in Medical Imaging (EP/L015226/1), jointly hosted between King's College, London, and Imperial College, London. The work was sponsored by GlaxoSmithKline (GSK), an industrial partner to the CDT. All contributions were authored at the Biomedical Image Analysis Group (BioMedIA) group in the Department of Computing at Imperial College London. The research was conducted under the supervision of Dr. Ben Glocker, Reader in Machine Learning for Medical Imaging from BioMedIA at Imperial College London. Secondary supervision was conducted by Prof. Daniel Rueckert, Head of Computing at Imperial college London. Industrial supervision was provided by Dr. Chris Page and Dr. Marius de Groot from GSK.

This work would not have been possible without access to the data provided by the UK Biobank (UKBB) imaging study under Access Applications 18545, 12579 and 2964, nor the manual contours of Cardiac Magnetic Resonance Imaging (CMR) scans contributed by Access Application 2964. The manual Quality Control (QC) scores contributed by H. Suzuki are also thankfully acknowledged.

The credit for the development of Reverse Classification Accuracy (RCA) lies firmly with Dr. V. Valindria *et al.* [164]. Credit for the base Image and Spatial Transformer Network (ISTN) framework belongs to Dr. M. Lee *et al.* [99].

Abstract

In medical image analysis pipelines, Machine Learning (ML) is increasingly employed to process images and to derive clinically relevant biomarkers and quantitative measures. Results of these analyses are used to inform clinical decision-making from diagnosis to treatment planning. It is imperative that ML models work reliably, and their results are adequately scrutinized to avoid introducing bias or inaccuracy into downstream tasks.

Image segmentation is a key component of many medical image analysis pipeline. For a segmentor to be reliable, it must be possible to identify instances when it has failed. This work presents an extensive validation of an approach for automated Quality Control (QC) of Cardiac Magnetic Resonance Imaging (CMR) segmentations using a reverse testing strategy, which is modified from Reverse Classification Accuracy (RCA) [164]. The method is evaluated on data from the UKBB imaging study, demonstrating its effectiveness at large scale in the absence of Ground Truth (GT). It achieves over 95% accuracy classifying segmentation quality using Dice Similarity Coefficient (DSC).

It may be desirable to obtain an immediate assessment of segmentation quality in high-throughput pipelines, in pharmaceutical trials, or in the clinic to inform operators when acquired images yield poor quality analysis results. This study presents a method for real-time automated QC using Deep Learning (DL)-based frameworks which directly predict DSCs and proxy RCA scores, achieving MAE = 0.03 and MAE = 0.14 respectively.

A model may fail because it is evaluated on data drawn from a different distribution to that on which it was trained, as with scans acquired from different sites. This work introduces and evaluates an unpaired, unsupervised method for Domain Adaptation (DA) on multi-site neuroimaging data, aiming to mitigate the degradation in model performance due to Domain Shift (DS) caused by population and acquisition changes. Over 20% performance is recovered in a classification task when applying constrained appearance and spatial transformations with Image and Spatial Transformer Networks (ISTNs).

Contents

List of Figures

List of Tables

List of Source Code

Acronyms

AAM Active Appearance Model. 64, 67
AF Atlas Forest. 97
AI Artificial Intelligence. 84
ANN Artificial Neural Network. 16, 39–42
AR Augmented Reality. 32
ASM Active Shape Model. 64
AUC Area Under Curve. 54
AUROC Area Under the Receiver Operating Characteristic. 54

BioMedIA Biomedical Image Analysis Group. 8
BN Batch-normalization. 47, 124, 147

Cam-CAN Cambridge Centre for Ageing and Neuroscience. 136, 153, 154, 158, 168–174, 176, 188
CC Correlation Coefficient. 72, 75
CDT Centre for Doctoral Training. 8
CMR Cardiac Magnetic Resonance Imaging. 8, 10, 13, 33, 88, 91–93, 96, 99, 102–104, 107–111, 113, 115, 118, 120, 121, 123, 131, 132, 181, 182, 186, 187
CNN Convolutional Neural Network. 15, 16, 18, 39, 44–49, 67, 68, 92, 97, 102, 104, 111, 112, 115, 118–123, 125, 126, 129, 131–133, 147, 181–183, 186, 188, 189
CoG Center of Gravity. 72, 73
CoM Center of Mass. 71, 99
CP Control Point. 70–72, 77, 82, 147, 155, 158, 159, 164, 165, 172–175
CPU Central Processing Unit. 120
CRF Conditional Random Field. 68
CT Computed Tomography. 33, 50, 64, 67, 69, 76, 84, 140
CV Computer Vision. 25, 30, 32, 33, 43

DA Domain Adaptation. 10, 15, 18, 24, 25, 27, 39, 136–139, 141–146, 148, 152, 159, 162, 164, 171, 172, 174, 175, 178, 180, 181, 184, 186, 188, 189
DH Data Harmonization. 27, 137, 184
DL Deep Learning. 10, 27, 28, 30, 31, 33, 39, 40, 49, 55, 56, 68, 91, 92, 104, 111, 115, 118, 120, 121, 123, 133, 145, 184, 189
DM Domain Mapping. 139, 142, 143
DNN Deep Neural Network. 40, 43
DS Domain Shift. 10, 24–28, 85, 90, 134, 136–140, 142–145, 148, 155, 158, 159, 162, 164, 175, 178–181, 184, 185, 188, 189
DSC Dice Similarity Coefficient. 10, 14, 16, 18, 27, 53, 90, 97–99, 103, 106–114, 116, 118, 120, 122–130, 132, 133, 182, 183, 186, 187

ED End-Diastolic. 102, 103, 123
EM Expectation Maximization. 65
EPSRC Engineering and Physical Science Research Council. 8

FC Fully-connected. 48, 68, 124, 147, 158
FCN Fully-convolutional Network. 46
FCNN Fully-connected Neural Network. 40, 43, 44, 46, 47
FN False Negative. 52, 94, 164
FOV Field-of-view. 25, 104, 121, 154
FP False Positive. 52, 94, 98
FPR False Positive Rate. 54, 103, 107, 110–112, 127, 129, 130, 183
FT Fourier Transform. 72

GAN Generative Adversarial Network. 49, 148, 179
GMM Gaussian Mixture Model. 16, 19, 66–68, 207
GPU Graphics Processing Unit. 120, 125
GSK GlaxoSmithKline. 8
GT Ground Truth. 10, 13, 16, 24, 27, 28, 33, 40, 49, 52, 53, 64, 67, 68, 84, 89, 90, 92–99, 102–113, 118–123, 125–129, 131, 133, 168–170, 175, 180–183, 185
GWAS Genome-wide Association Study. 102, 104

HD Hausdorff Distance. 99–101, 107–113

I2I Image-to-Image. 30, 31, 49, 50, 139, 143, 146–149, 184
IG Information Gain. 37
IoU Intersection over Union. 121, 122
IQA Image Quality Assessment. 92, 121
ISTN Image and Spatial Transformer Network. 8, 10, 15, 17, 18, 27, 136, 143–151, 155–160, 162–167, 171–181, 184, 186, 188, 189
ITN Image Transformer Network. 18, 146–148, 155, 157–160, 162–165, 167, 172–174, 176, 177, 184, 188

JCMR Journal of Cardiovascular Magnetic Resonance. 6, 88

K-S Kolmogorov-Smirnov. 77

LCC Linear Correlation Coefficient. 121
LR Learning Rate. 43
LSTM Long Short-term Memory. 40
LVC Left Ventricular Cavity. 26, 85, 99, 103, 106–112, 123, 125, 128, 131, 182
LVM Left Ventricular Myocardium. 26, 99, 103–114, 123, 125, 127–129, 131, 132, 182, 187

MAE Mean Absolute Error. 16, 18, 107–112, 118, 126–131, 133, 168–170, 174, 175, 182, 185
MEG Magnetoencephalography. 154
MI Mutual Information. 64, 65, 74–76, 189, 213
MICCAI Medical Image Computing and Computer Assisted Intervention. 6, 88, 118, 136
ML Machine Learning. 10, 15, 24–28, 30, 31, 33, 34, 40, 52, 55, 56, 66, 84–87, 92, 94, 97, 134, 136, 137, 140, 141, 180, 181, 184, 186, 189
MLP Multilayer Perceptron. 40, 44, 48
MNI Montreal Neurological Institute. 154
MR Magnetic Resonance. 76, 85, 140, 179
MRC Medical Research Council. 154
MRI Magnetic Resonance Imaging. 15, 17, 18, 26, 32, 33, 50, 57, 60, 61, 64, 67, 69, 76, 80, 84–87, 92, 96, 106, 136, 140, 144, 146, 148, 152–155, 158–161, 164, 168–171, 173–175, 181, 182, 188, 189

MSD Mean Surface Distance. 99–101, 107–113
MSE Mean Squared Error. 40, 43, 125, 168, 169, 179

NLP Natural Language Processing. 25
NN Nearst-neighbour. 81, 82

OCR Optical Character Recognition. 32
OHDSI Observational Health Data Sciences and Informatics. 87

PDE Partial Differential Equation. 60
PDM Point Distribution Model. 64
PET Positron Emission Tomography. 69, 76
PPMCC Pearson Product-Moment Correlation Coefficient. 75
PPV Positive Predictive Value. 53
PR Precision-Recall. 54

QA Quality Assurance. 85
QC Quality Control. 8, 10, 13–16, 26–28, 30, 31, 34, 39, 84–86, 88, 89, 91–94, 96, 104, 105, 113–115, 118–123, 125, 126, 129, 131, 132, 180–183, 186–189
QI Quality Indicator. 122

RBF Radial Basis Function. 82
RCA Reverse Classification Accuracy. 8, 10, 14–16, 18, 27, 88, 91, 95–99, 102–117, 119, 120, 123, 126, 127, 129–133, 180–183, 186–189
ReLU Rectified Linear Unit. 41, 42, 124, 147, 158
ResNet Residual Network. 124, 143
RF Random Forest. 16, 18, 36, 39, 67, 68, 92, 102–104, 107–111, 113, 119, 123, 124, 131–133, 181–183, 186, 188
RL Reinforcement Learning. 33, 34
RMS Root Mean Squared Error. 99–101, 107–113
RNN Recurrent Neural Network. 40
ROC Receiver Operating Characteristic. 54
RVC Right Ventricular Cavity. 99, 103, 106–112, 123, 125, 128, 131, 182
RVM Right Ventricular Myocardium. 99

SAX Short-Axis. 99, 102, 103, 123
SSIM Structural Similarity Index. 172, 174, 176, 182, 189
STAPLE Simultaneous Truth and Performance Level Estimation. 65
STN Spatial Transformer Network. 18, 146–148, 155, 157–160, 162–165, 167, 171–177, 184
SVC Support Vector Clustering. 36
SVM Support Vector Machine. 33, 35, 36, 65–68, 92, 94

TA Acquisition Time. 154
TE Echo Time. 154
TI Inversion Time. 154
TL Transfer Learning. 139
TN True Negative. 52
TNR True Negative Rate. 53
TP True Positive. 52
TPR True Positive Rate. 53, 54, 103, 107, 110–112, 127, 130, 183

TR Repetition Time. 154

UKBB UK Biobank. 8, 10, 13, 18, 25, 27, 28, 85–89, 92, 93, 96, 102–104, 110–113, 119–123, 136, 137, 153, 154, 158, 168–174, 176, 188

VAE Variational Autoencoder. 122

WH Whole-Heart. 99, 106–108, 110–112, 123, 125, 127–132, 182, 187

Chapter 1

Introduction

Many medical image analysis pipelines make use of Machine Learning (ML) models to acquire, reconstruct, process and analyze imaging data. A pipeline many be composed of many modules which perform tasks such as image registration or segmentation. Each analysis step has the potential to introduce bias and errors to the data which may affect subsequent analyses. The predictions from such ML models are used in clinical decision making, therefore it is imperative that any model deployed in a medical image analysis pipeline is thoroughly evaluated in order to ensure safety and gain clinical trust.

To ensure the reliability of a ML model, the quality of its output may be assessed. In this work, particular attention is paid to the quality of automated image segmentations. Segmentations may be automatically processed to derive clinically relevant biomarkers from images which can be used to inform diagnosis and treatment planning. It is also necessary to have an objective measure of quality to reduce inter- and intra-rater variability. If the analysis is to be performed at large scale, an automated method would be preferred. Such quality measures must be able to work in the absence of Ground Truth (GT) labelmaps to avoid laborious manual image annotation. Where immediate feedback on segmentation quality is required, a method that works in real-time is needed.

Where the data on which a ML model is to be evaluated is drawn from a different distribution than that on which it was trained, a degradation in model performance may be seen. This may occur due to differences in hardware, acquisition protocols or the population of subjects being scanned. The Domain Shift (DS) problem may be mitigated by performing Domain Adaptation (DA) in order to harmonize datasets, but many of these methods lack explainability. An approach using constrained appearance and spatial transformations may offer a more intuitive indication of the domain mapping.

1.1 Overview.

Machine Learning (ML) has become ubiquitous in every day life. Even handheld devices with limited processing power now benefit from research in Natural Language Processing (NLP), facilitating the rise of virtual assistants[1], and advances in Computer Vision (CV), improving the quality of photographs [167] and video streams. The models grounded in CV are the focus of this work.

The imaging data acquired by medical devices can be very different to those captured by a smartphone camera. The medical image acquisition process may be highly complex requiring large and expensive pieces of machinery. Computation of the optimal acquisition parameters needs careful attention by trained technicians. Infirm, elderly or particularly young subjects may struggle to perform the actions necessary to obtain good quality images, for example breath-holding. Image reconstruction from raw data is not straightforward and benefits from active research in order to reduce the scanning time and improve image quality [43, 104]. The resulting images are highly heterogeneous, varying by modality, vendor [100], device, protocol, operator, and reconstruction and processing algorithms. In the clinical setting, each scan is acquired according to some particular clinical need, and the individual needs of the patient, leading to differing positions and Fields of View (FOV).

ML models require large datasets from which to learn. The training dataset should be representative of the data on which the model will be evaluated. Formally, the training and testing datasets should be drawn from the same distribution. However, with such heterogeneity in medical imaging data, obtaining multiple independently acquired datasets from the same distribution, or 'domain', is challenging. Large-scale national initiatives like the UK Biobank (UKBB) [154] aim to acquire data at the population scale using standardized hardware, operators and acquisition protocols [123].

Whilst large-scale imaging centers may aim to acquire, relatively, consistent data, the fact remains that ML models trained on this dataset will likely fail to perform as well when evaluated on images from any other site or subject population. This difference is referred to as Domain Shift (DS) [56] when moving from one data distribution to another. Options to mitigate the effects of DS tend to fall into two groups: altering the source or target data distributions such that they match, or modifying the ML model to be agnostic to domain differences.

Methods that aim to improve the robustness of ML models to changes in domain, fall under the purview of Domain Adaptation (DA). Approaches that deal with domain-invariant feature learning [83] often lack the explainability of those which work to harmonize the data input. However, recent methods that aim to align training and test distributions often apply transformations wholesale using some difficult-to-interrogate ML model. In the medical imaging case, it may be desirable to apply transformations which are constrained to remove variations of appearance and subtle spatial

[1] https://assistant.google.com/

differences that can be better visualized and explained, and that are anatomically plausible.

Whichever road is taken, the overriding idea is to strengthen the reliability of ML models. In many cases, the consequences of poor model generalizability are not serious, *e.g.* a misaligned facial filter on a social media post. However, in the medical imaging domain, ML models are used to inform clinical decision making that may determine the diagnosis and treatment of a patient. Thus, the reliability of any model that may be deployed in the clinical setting must be heavily scrutinized in order to garner the trust of clinicians and patients [165].

The pipeline through image acquisition, reconstruction, processing and analysis may be composed of many modules that perform some transformation on the imaging data or provide some measurement ready for subsequent analysis. Each of these modules has the potential to introduce some bias or error to the data [150], so it is important that each part is carefully tested. Such tests are usually performed before deployment often by the researchers and developers by whom the module was constructed. But, it is also necessary to carry out routine or even continuous appraisal of the quality of the outputs from each component in the image analysis pipeline. Vigilant Quality Control (QC) may help to detect occasions when the components fail and qualify the reasons for such failure, *e.g.* Domain Shift.

One such component of the image analysis pipeline is a segmentation model. A segmentation is a voxel-wise classification that assigns class labels to each element of the image in order to partition it into semantically meaningful regions. For example, segmentations of medical images may extract organs from full-body Magnetic Resonance Imaging (MRI) [164]; delineate boundaries between the Left Ventricular Myocardium (LVM) and Left Ventricular Cavity (LVC) in cardiac scans [9]; or locate and quantify brain tumours [38]. Such segmentations or 'labelmaps' are useful for extracting clinically relevant information such as blood volume or vessel thickness. If a segmentation model fails to produce a good quality segmentation, then the subsequent analysis will be flawed. It is crucial to be able to detect these instances where a segmentation has failed.

Quality Control (QC) of image segmentations generated by an automated model may be done by visual inspection conducted by an expert. Multiple raters may be used in order to combat intra-rater variation, though inter-rater variability may also exist. A repeatable and reproducible objective method for assessing the quality of automated segmentations would therefore be beneficial. An automated method for segmentation QC would sit as a complementary module in the image analysis pipeline and would flag low quality results. Such a method could also be employed for segmentations of large-scale datasets where manual QC is infeasible.

Moreover, an automated approach to segmentation QC that acted in real-time could allow instantaneous feedback to the segmentation module. A poor quality result from one algorithm could force the module to try another, or perhaps trigger some retraining or fine-tuning of the model. Instantaneous feedback would also be beneficial to the operator in the acquisition setting, alerting them to the poor quality of a segmentation result and allowing them to reacquire images if necessary

whilst the subject is still within the scanner.

The reliability of ML models requires assessments of their inputs and outputs. It is important to ensure that the data for training and testing purposes is harmonized, and that the output from the model is of good enough quality to be used in downstream tasks and subsequent analysis.

1.2 Contributions.

This work is focused on the reliability of ML models in medical image analysis. First, it considers the quality of the outputs generated by ML models, with a focus on biomedical image segmentation QC at large scale. It moves to study real-time approaches to segmentation QC which may be desirable in the clinic or in high-throughput pharmaceutical trials. The work finishes by studying the DS problem on model inputs, to ensure consistency in model performance across domains. The three areas of contribution are summarized here.

1. Large-scale Automated QC in the Absence of Ground Truth (GT).

A validation of a reverse testing strategy, based on Reverse Classification Accuracy (RCA), for the automated QC of biomedical image segmentations. Importantly, it demonstrates that a modified RCA can accurately assess the quality of automated segmentations on a *per-instance* basis in the absence of GT. The method is evaluated at large-scale with data from the UKBB study and provides quality measures in terms of common, well-understood metrics.

2. Real-time Automated QC.

An extension to RCA is proposed which learns to predict RCA scores in real-time. It shows that the Dice Similarity Coefficient (DSC) can be directly predicted from an image-segmentation pair by training a Deep Learning (DL)-based model on a large number of generated segmentations of varying quality. It provides a framework by which QC predictions can be made, at large scale, in the absence of GT and in real-time by employing RCA to create labels for training data.

3. Data Harmonization (DH) via Constrained Image and Spatial Transformations.

Finally, this work moves to consider the inputs to ML models by considering the DS problem. It aims to show that a model trained on one domain can be retrained, or finetuned, in order to recover the significant drop in performance when evaluated on a new domain. Specifically, this work contributes the idea that unsupervised and unpaired DA can be achieved through constrained appearance and spatial transformations enforced by Image and Spatial Transformer Networks (ISTNs) which are trained adversarially. Such methods may reduce the effects of DS in medical imaging by diminishing some of the heterogeneity in the data.

Chapter 2 provides the technical foundation for the concepts encountered in this work. It discusses the fundamentals of predictive modelling and describes some of the common ML algorithms used for classification and regression problems. The essentials of DL, including generative modelling, are explored as the basis for the models employed in this work. Image segmentation is considered, and common techniques for pixel-wise classification are outlined. The process of image registration is then detailed which is important for the approach to QC utilized in this work. Finally, the challenges associated with deploying ML models in the medical imaging domain are reviewed.

The contributions of this work, as outlined above, are detailed in subsequent chapters. Chapter 3 presents an approach to automated segmentation QC at large scale in the absence of GT. It is extensively validated using data from the UKBB population imaging study. This work is extended in chapter 4 with a DL-based framework to perform segmentation QC in real-time when the GT is unavailable. Finally, chapter 5 considers the DS problem with a focus on neuroimaging data acquired in two different domains. It aims to show that constrained image and spatial transformations can be applied to the data in order to recover the degradation in model performance when switching between domains.

Finally, the primary conclusions and limitations of this work are discussed, and the directions for future research are laid out.

Chapter 2

Background

The basis of this work lies in predictive modelling, whereby a model is defined and its parameters learned such that it can accurately predict some outcome given an input. In this chapter, Machine Learning (ML) techniques for approaching classification and regression tasks are described and the fundamentals of Deep Learning (DL) are explored. Generative modelling, including Image-to-Image (I2I) translation, is described.

This work relies on innovations in Computer Vision (CV) and image processing that are under constant improvement by active research. The chapter outlines techniques for partitioning an image into salient regions, known as image segmentation, including traditional, registration-based, and ML-based methods. Image registration is employed in this work to align large numbers of images such that further analyses can be performed. The process of image registration, and medical application, is explained in this chapter.

Finally, the challenges associated with deploying ML models in the field of medical imaging are considered. Issues around Quality Control (QC) and bias in ML are explored.

This chapter identifies and describes the relevant technical details for the methods to be employed in the contribution chapters of this work. The content of this chapter is not intended to be exhaustive, but rather *indicative* of the most common techniques relating to the fundamental concepts that will be considered in the contribution chapters of this work.

2.1 Overview.

This chapter aims to supply the technical knowledge that provides a foundation to the contributions of the works in chapter 3, chapter 4, and chapter 5. This includes:

Section 2.1. Predictive Modelling.

Methods of predicting some outcome given an input with a particular focus on classification and regression problems in medical imaging. Fundamentals of Deep Learning (DL) are considered and more complex Image-to-Image (I2I) translation frameworks are discussed.

Section 2.2. Image Segmentation.

The process of partitioning an image into salient regions. Traditional methods, registration-based approaches and Machine Learning (ML) techniques are considered in this section.

Section 2.3. Image Registration.

The process of aligning two images. The registration pipeline is outlined including feature extraction, feature matching, transformation estimation and image resampling. The importance of interpolation is also considered here.

Section 2.4. Deployment of ML Models in Medical Imaging.

The challenges faced when ML models are deployed in real-world scenarios are discussed. The focus is on Quality Control (QC) and bias in ML.

2.2 Predictive Modelling.

2.2.1 Introduction.

When the outcome of an experiment can be predicted based on some data input, the process can be modelled such that given a new set of data, the model will provide a probable outcome. Models can be categorised based on the kinds of predictions they are trying to make.

Classification. A model whose output indicates some measure of belonging to a particular category, group or class is referred to as a 'classifier'. The prediction on some input may be a single scalar integer value denoting to which class the input belongs. It is also possible to predict a scalar or vector of floating-point values indicating probabilities of the input belonging to different groups. Predictions place the input into classes based on their similarity, or dissimilarity, to training samples. In Computer Vision (CV), predictions on the input can be made at the global scale, as in image classification, or pixel-level, as in image-segmentation.

Regression. In contrast, a regressor is a model whose aim is to predict a continuous value based on the input. The model does not offer an output based on discrete groups or predefined clusters, but instead aims to predict an individual continuous value outcome for each input. The prediction does not indicate 'belonging', but rather some real value which positions the input in relation to other training samples. Regression also includes synthesis tasks where a completely new output is generated based on the parameters of the input.

This work is grounded in the field of CV where the aim is to predict some output when the input is a digital image. Thus classification models may group images based on their subject [42], for example, separating a set of photographs based on the type of animal present in the image, or identifying particular breeds. Classification problems in CV also involve object detection, *i.e.* determining whether a particular class is present in an image. CV-based classification models have been used for a long time by postal services to read handwritten characters with Optical Character Recognition (OCR) [97].

Regression tasks in CV can be complex, such as estimating the angle made between joints in body pose estimation [163] for Augmented Reality (AR), or, controversially, face recognition [113]. It is used in object localization [55] within an image. Synthetic images can also be produced by regression tasks [58] which may be useful for reducing the blur in photographs or changing the style of the input image. Regression is used in object tracking [59], and in recent years, to add filters to photos and live streams [33].

The work presented here falls, more specifically, under the domain of biomedical image analysis, where the images under test are those acquired by medical imaging scanners such as Magnetic

Resonance Imaging (MRI) or Computed Tomography (CT). Classification tasks in medical image analysis include disease diagnosis [65], where images are classified as 'healthy' or 'pathological' based on the presence of disease indicators. Segmentation is a classification task where structures, *e.g.* tumours [38], are separated in the image and passed for subsequent analysis.

In terms of regression, age-prediction from brain MRI [85] can aid in the diagnosis of dementia. In Cardiac Magnetic Resonance Imaging (CMR) and echocardiography, regression can be used to automatically determine physiological metrics such as pressure within blood vessels [181].

Classification and regression need not be considered independently; they've been successfully combined in surgical settings guiding surgeons, and robots, via instrument-tracking [127], and aiding technicians acquiring images by identifying imaging planes [20] and suggesting next steps.

Many advances have been made which improve the performance of classification and regression models in the medical imaging domain. The methods applied to medical images primarily originate from CV work in the natural image domain and increasingly involve Machine Learning (ML).

2.2.2 Machine Learning.

ML is a generic term covering many predictive modelling methods which involve some form of iterative update step to the model's parameters. The information used to perform the update comes from a set of training data over which some metrics are calculated by the model. These metrics form an objective function, also known as the cost or loss, which must be optimized by the algorithm. Broadly, ML algorithms fall into two categories depending upon the availability of desired target labels in the training dataset: supervised or unsupervised learning.

Supervised Learning. In supervised learning, it is assumed that the training dataset contains labels which indicate the result that the model should aim to predict during training. The updates to model parameters depend on the error between the model's current prediction and the expected Ground Truth (GT) label from the training set. Supervised ML algorithms include Support Vector Machines (SVMs) and many Deep Learning (DL)-based approaches. Classification and regression are supervised learning problems.

Unsupervised Learning. In unsupervised learning, no labels are available in the training set and so the model must learn based only on information that it can gather from the data themselves. For images, this may be image element intensities, local textures, or shape features. Unsupervised ML algorithms include clustering methods. These techniques may also be used in *semi-supervised* learning, where only a few target labels are available, but the majority of the data remains unlabelled.

Other branches of ML exist, such as Reinforcement Learning (RL), where some 'agent' learns to

operate within an 'environment' through a feedback mechanism in order to maximize some reward. Such methods have been used to teach agents how to play games [109] and for medical image segmentation [141]. However, RL is not directly applied in this work.

The work presented here considers ML methods for biomedical image segmentation and their potential use for automated Quality Control (QC) of such segmentations. ML methods are also considered as possible ways to ensure the reliability of models when employed on differing datasets.

This section aims to relate common ML frameworks which may be used in classification problems such as segmentation and regression problems like age prediction. First, some common traditional ML methods are laid out and those which are employed directly in this work are described. Section 2.3 on page 56 details how such algorithms may be used specifically for the purpose of segmentation.

2.2.2.1 K-means Clustering Method.

K-means is a generic unsupervised data clustering algorithm that sorts data into K classes based on their 'distances' from the cluster centroid. More specifically, it aims to reduce the intra-class or intra-cluster variance over all samples.

First, K seed points are selected, either randomly or following some pre-defined criteria. The distance to the centroids is calculated for all samples in the dataset. Each sample is assigned to the same group or class as its closest centroid. The mean of each cluster in K is recalculated and the process is repeated until the cluster means stop changing - with respect to some threshold - or some number of iterations is reached.

Commonly, the sum of squared differences shown in equation (2.1) is iteratively minimized where K is the number of clusters; N is the number of samples assigned to cluster j; x_i^j is the i^{th} sample in cluster j; and c_j is the centroid of cluster j.

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} \left|\left| x_i^j - c_j \right|\right|^2 \tag{2.1}$$

The number of clusters is difficult to estimate implicitly and is usually specified by some external criteria, *e.g.* if the number of classes is already known. The number of clusters could be set as the number of training samples and then decreased based on some similarity criteria between samples. The K-means algorithm is costly as it iterates over every sample on each pass.

An alternative to this hard-clustering method is to use soft-clustering through soft K-means or 'C-means'. C-means is a 'fuzzy' clustering method which uses a similar approach to K-means. Fuzzy clustering methods assigns each sample K values, one for each class or cluster. Each value represents the probability of belonging to cluster $k \in K$. Equation (2.2) on the following page

shows the objective function for this algorithm which includes the membership term u_{ij}^m indicating the degree to which a sample x_i belongs to cluster c_j and $m \in \mathbb{R} > 1.0$ is the cluster fuzziness [69].

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} u_{ij}^m \left|\left| x_i^j - c_j \right|\right|^2 \tag{2.2}$$

$$u_{ij}^m = \frac{1}{\sum_{l-1}^{K} \left(\frac{|x_i - cj|}{|x_i - ck|} \right)^{\frac{2}{m-1}}} \tag{2.3}$$

The cluster centroids are then the mean of all points weighted by the degree of cluster membership as in equation (2.4).

$$C_j = \frac{\sum_{i=1}^{N} u_{ij}^m x}{\sum_{i=1}^{N} u_{ij}^m} \tag{2.4}$$

This method may be useful for circumstances where it is important to get some idea of the confidence of a sample per class which can support clinical decision making processes [175].

2.2.2.2 Support Vector Machine Classification Method.

A simplistic supervised implementation of a Support Vector Machine (SVM) [35] creates a non-probabilistic binary linear classifier which separates datapoints from two classes by some linear boundary or 'hyperplane'. The hyperplane is chosen such that it maximizes the separating margin between the nearest datapoints, known as *support vectors*, on either side. Support vectors indicate the domain of support for this hard margin. Larger margins reduce the generalization error of the model. The hard separating margin is not always guaranteed to exist and some samples may fall across the class boundary creating a soft margin.

For a set of n training datapoints $\{(\mathbf{x}_i, y_i)\}_1^n$, where $\mathbf{x}_i$ are feature vectors and $y_i \in \{-1, 1\}$ are binary class labels, any hyperplane can be written as $\mathbf{w} \cdot \mathbf{x} - b = 0$ where $\mathbf{w}$ represents the normal to the hyperplane and $b/\left|\left|\mathbf{w}\right|\right|$ gives the offset of the hyperplane from the origin along $\mathbf{w}$. The model aims to avoid datapoints falling into the separating margin by enforcing the constraints in equation (2.5) and equation (2.6). The prediction is given by equation (2.7).

$$\mathbf{w} \cdot \mathbf{x_i} - b \geq 1, \forall\, y_i = 1 \tag{2.5}$$

$$\mathbf{w} \cdot \mathbf{x_i} - b \leq -1, \forall\, y_i = -1 \tag{2.6}$$

$$y(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \forall\, 1 \leq i \leq n \tag{2.7}$$

The model optimization problem is thus to minimize $\left|\left|\mathbf{w}\right|\right|$ subject to equation (2.7).

This formulation creates a hard-margin where the data are linearly separable. A soft-margin formulation exists for the converse which takes into account the distance from the margin of datapoints lying on the wrong side.

In image analysis, SVMs can realise non-linear discriminations between classes by using the kernel trick [95] to transform the classification problem into higher-dimensional feature-space where separating hyperplanes can be found.

An unsupervised, and widely used, variation of SVM was introduced in [11]. In Support Vector Clustering (SVC), the data is mapped to a higher-dimensional feature space using a Gaussian kernel. The minimal enclosing sphere is then found which, when mapped back to data space, separates into multiple components that each enclose clusters of points. The samples can then be classified.

2.2.2.3 Decision Tree Methods.

In this work, Random Forests (RFs) are employed for biomedical image segmentation which is an image classification task at the pixel-level. The algorithm is set out here.

A Decision Trees.

Decision trees [136] are supervised models that continually separate samples into smaller and smaller groups based on some binary decision making process. The groups are split until they contain members of only a single class or value for classification and regression trees respectively. Decision trees can be used for classification or regression problems where the target variables are discrete or continuous values respectively.

A tree comprises a root node, internal nodes, and leaf nodes where each root and internal node is connected to two other nodes by branches. Each node separates the incoming data samples before passing them to its two child nodes. Each leaf node is labelled to indicate that the samples occupying the node belong a particular class or have a particular value. The splitting process stops at the leaf nodes.

Splitting is done such that the resulting sets are as homogeneous as possible: each set contains as many samples of one class as possible or samples that have similar values. The result is that successive 'decisions' produce groups that contain more samples from fewer classes increasing the purity of the set. To achieve this, a selection criterion is chosen, such as entropy.

Entropy can be seen as a measure of impurity or randomness in a dataset. The Shannon entropy [149] $H(X)$ for the random variable X, with $k \in K$ discrete states is shown in equation (2.8) on the following page. A 100% pure dataset which contains samples belonging only to a single class

has $H(X) = 0$. The maximum entropy value $\log_2(|K|)$ depends on the number of classes $|K|$.

Shannon Entropy:
$$H(X) = -\sum_k p_k(X = k) \log_2 p_k(X = k) \tag{2.8}$$

Nodes perform initial splits using each feature in the sample and calculate the entropy for each. The Information Gain (IG) [37] in equation (2.9) is the difference in entropy calculated before and after the split where α is the splitting criterion. The feature that splits the data into sets that give the greatest IG is selected as the decision criterion for the node. The first decision that is made uses all of the data and is done at the root node. Intuitively, the feature that is chosen in this first splitting must be the most discriminative, or the most important. When the entropy of the sets in a node cannot be reduced any further, they are pure and the splitting process stops. A depth D can be specified such that the splitting process stops after D levels are created.

Information Gain:
$$\text{IG}(X, \alpha) = H(X) - H(X \mid \alpha) \tag{2.9}$$

Alternatively, the Gini impurity [136] shown in equation (2.10) may be used to inform the splitting. It is calculated by subtracting the sum of squared probabilities of each class from one. The feature giving the lowest Gini index is chosen for the split. This approach can favour larger partitions than IG.

Gini Index:
$$I_G(X) = 1 - \sum_k p_k(X = k)^2 \tag{2.10}$$

For regression trees, the mean label value of the samples within a set is taken as the prediction for the whole group. The sum of squared errors in equation (2.11) is calculated for each possible split over all features. The feature that creates sets which minimizes the sum is chosen as the decision criterion for the node.
$$\text{SSE} = \sum_k (y - \hat{y})^2 \tag{2.11}$$

During inference, samples start at the root node and follow the decision making process that was used during training. They are assigned the value or class label of the leaf node at which they end.

Decision trees scale well to datasets with large numbers of features, by increasing the number of decisions, or nodes. However, trees that become very deep tend to overfit their training sets and thus have very high variance. They are explainable in that their decisions can be interrogated at each node, which is important in some applications such as medical image analysis.

B Pruning.

The overfitting problem with decision trees can be partially addressed with pruning. Pruning is the process of removing leaves or entire sub-trees from a trained decision tree in an effort to improve the generalizability of the model. Instead of potentially ending up with a single training example per leaf node, the leaves may now contain examples from different classes and therefore some distribution is seen. The decision on when to use pruning may be task dependent.

C Boosted Trees.

Boosting [143] is a method of combining a number of weak learners into a single, stronger model. A set of trees T is created sequentially where each tree $t \in T$ is given a weight w_t based on the accuracy of its output $t(x)$ on samples x. Before each successive weak learner is trained, the samples are weighted such that those which were misclassified are given a larger weighting. Therefore, future weak learners focus on previously misclassified examples. The ensemble prediction for a sample is given in equation (2.12) by summing the weighted contributions from each weak learner.

The adaptive boosting, or 'Adaboost' [51], algorithm was one of the first such approaches. It learns decision stumps, *i.e.* decision trees with a single split, which are combined into a strong classifier. It is 'adaptive' in that it modifies subsequent weak learners to prioritize misclassified examples.

$$\hat{y}(x) = \sum_{t} w_t t(x) \tag{2.12}$$

D Bootstrap Aggregation.

Bootstrap aggregation [17], or 'bagging', is another method that uses an ensemble of models to form its final prediction. Instead of learning many weak learners using the same data, the samples used to train each learner are randomly sampled from the dataset with replacement. Thus, each model is trained on a different subset of the original data. The prediction of a bagged model is the consensus, a majority vote, amongst all of the trained learners in the classification case, or the mean of the predictions for regression.

The trees generated with bagging are less correlated than if each were trained on the same data. As such, the algorithm is less sensitive to noise in the training set than are the individual trees. As with boosting, bagging reduces the variance of the overall model.

E Random Forests.

Bagging deals with the random selection of the samples used for training each learner in an ensemble, but the remainder of the decision tree algorithm remains the same. In random decision forests, [73], the algorithm is modified such that the features available to each node during the splitting process are also randomly selected. This is sometimes referred to as 'feature bagging'. If a small number of features are very strong predictors of the desired output, many of the trees

will select them. The idea of feature bagging is to de-correlate successive trees in the forest by limiting the visible features.

The term Random Forest (RF) [18] was later proposed, and trademarked, which combines the random selection of training data (bagging) and the random selection of features per node (feature bagging) into a single algorithm. Typically, for an input with p features, $p' = \lfloor \sqrt{p} \rfloor$ are selected at each node for classification and $5 \leq p' < \lfloor p/3 \rfloor$ for regression.

F Extremely Randomized Trees.

In 'ExtRa Trees', rather than calculating the optimal split at each node, a random value - sampled from the uniform distribution - is selected from the empirical range of each feature available at the node. The feature which best splits the set at its given random value is chosen as the splitting criterion. Feature bagging can still be applied in this approach, but the whole training set is used to train each tree in the forest rather than using bootstrap samples.

2.2.2.4 Deep-learning Methods.

The methods outlined above work well when the training data is well-structured and is labelled. However, imaging data can be highly heterogeneous and may lack labels if the dataset is very large. Though these methods learn which features are most discriminative for the given task, the features are simple, usually 1-dimensional, and often found exhaustively.

With large datasets, it is possible to employ Artificial Neural Networks (ANNs) which can have high capacity and learn highly complex features in multiple dimensions. ANNs benefit from the contributions of many individual decision makers which are weighted and combined. Each 'layer' of an ANN can be many nodes wide and may become very deep, having many layers. Input data can be 'encoded' into a higher level representation which gives the ANN more information about the training data as a whole.

DL-based methods commonly employ Convolutional Neural Networks (CNNs) for the purpose of biomedical image segmentation, but they are not explicitly employed for this task in this work. However, labelmaps generated by a DL-based algorithm are used as part of a validation experiment in chapter 3 on page 88 and further in chapter 4 on page 118. In this work, Convolutional Neural Networks (CNNs) are used to predict Quality Control (QC) scores as well as to perform Domain Adaptation (DA). The appropriate fundamentals for DL-prediction tasks as well as generative DL models for image synthesis are described in section 2.2.3 on the next page.

2.2.3 Deep Learning.

Deep Learning (DL) describes a framework in which data is passed through an ANN with a large number of layers. Such Deep Neural Networks (DNNs) pass data from the input layer to the output layer via many intermediate 'hidden' layers, each of which contains a series of 'neurons' or nodes. The nodes between layers are connected by 'weights' that indicate how much each node in the previous layer contributes to the value at the node in the current layer. In order to learn more complex functions, the nodes of the DNN also apply a non-linear 'activation' function to their outputs.

A DNN in which every node in one layer is connected to every node in the subsequent layer is called a Fully-connected Neural Network (FCNN). The Multilayer Perceptron (MLP) is an early FCNN. DNNs are typically 'feed-forward' networks, where the data flow only from input to output layers. Other variations exist such as Recurrent Neural Networks (RNNs) which allow the data to move in any direction, and are useful for giving the model a form of memory such as the Long Short-term Memory (LSTM) [74].

As with other ML algorithms, the DNN must be trained to achieve accurate predictions on the input at the output layer. Simple metrics such as the Mean Squared Error (MSE) between prediction and GT label are calculated to assess the current performance of the network. The error is used to modify the weights connecting the nodes of the model by 'back-propagating' [140] its gradient, layer-by-layer, to the input. At each layer, the weights connecting the next layer to the current nodes are updated. Thus, the network is iteratively optimized by computing the gradient of the output error with respect to the input data via the chain rule. Back-propagation refers to the method for computing this gradient, but the process of learning from this gradient is performed by algorithms such as stochastic gradient descent.

Formal definitions and derivations of back-propagation and neural network learning schemes are readily available from many sources, here only an overview of the concepts is provided.

A DNN is trained on the dataset $(X = \{x_1, ..., x_i\}, Y = \{y_1, ..., y_i\})$ where $x_i \in \mathbb{R}^d$ is a feature vector of $d = 3$ features, and $y_i \in \mathbb{R}$ are labels associated with each x_i. The DNN is a 3-layer network comprising input, hidden and output layers i, j and k respectively. The network is visualized in figure 2.1 on the following page.

The weights W_{ij} connect the nodes in layer i to those in layer j and can be defined in a matrix, as shown in equation (2.13). The number of rows indicates the number of inputs to j from i and the number of columns indicates the number of nodes in the current layer, j. In this example, there are 3 nodes in each.

$$W_{ij} = \begin{bmatrix} w_{11}^{ij} & w_{12}^{ij} \\ w_{21}^{ij} & w_{22}^{ij} \\ w_{31}^{ij} & w_{32}^{ij} \end{bmatrix} \tag{2.13}$$

The weighted input Z_j to the layer j is calculated by multiplying the layer inputs x_i with the weight

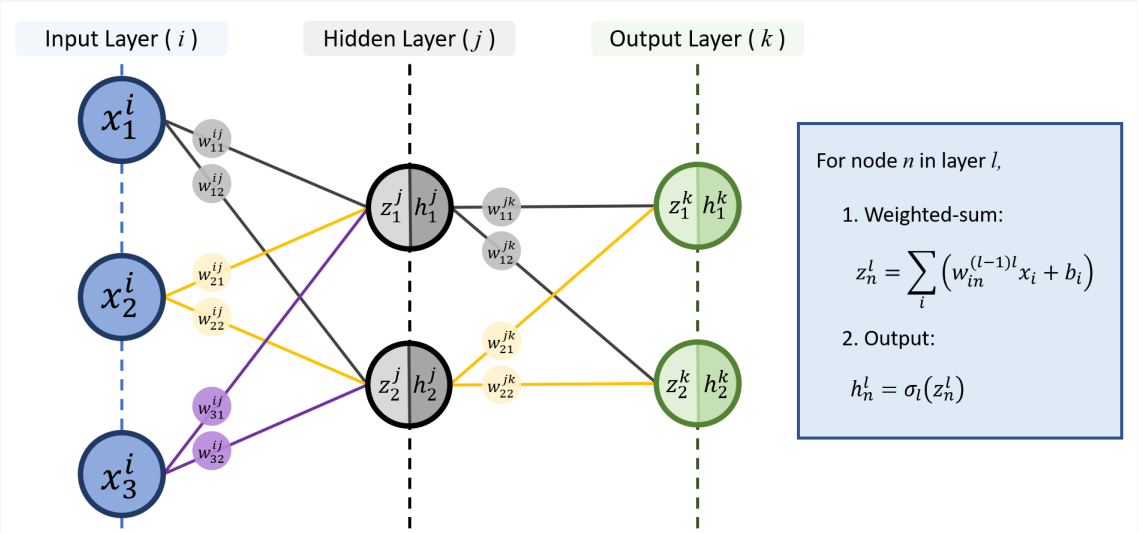


Figure 2.1: Visualization of a 3-layer Artificial Neural Network (ANN). The features x_i occupy the nodes of the input layer i. Each weight $w_{nm}^{(l-1)l}$ connects the n^{th} node in layer $(l-1)$ to the m^{th} node in layer l. The weighted sum entering the m^{th} node in layer l is z_m^l. The activation function σ_l applies to the weighted sum at each node in layer l giving output h_m^l.

matrix W_{ij} and adding a bias term B_j as in equation (2.15).

$$Z_j = x_i W_{ij} + B_j \tag{2.14}$$

$$\begin{bmatrix} z_1^j & z_2^j & z_3^j \end{bmatrix} = \begin{bmatrix} x_1^i & x_2^i & x_3^i \end{bmatrix} \times W_{ij} + \begin{bmatrix} b_1^j & b_2^j & b_3^j \end{bmatrix} \tag{2.15}$$

At each node in layer j, a non-linear activation function σ_j is applied. Common forms include the `Rectified Linear Unit (ReLU)` and `Leaky-ReLU` defined in equation (2.16) and equation (2.17) respectively. In classification problems, the activation function applied at the output layer depends on the number of classes to be predicted. For binary classification, the `sigmoid` activation may be used, shown in equation (2.18) on the next page. It can be extended to the multi-class case with K classes to achieve the `softmax` function in equation (2.20) on the following page. The `sigmoid` and `softmax` functions ensure that the output values are in the range $[0, 1]$ with all values summing to 1 in the `softmax` case. The $\tanh$ activation function can force outputs to the range $[-1, 1]$ instead. Graphical visualizations of the common activation used in this work are

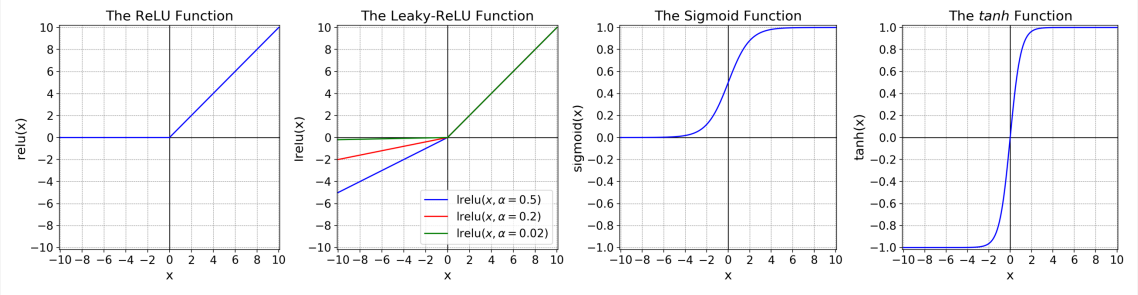


Figure 2.2: Common Activation Functions Applied in ANN. (left-to-right) Rectified Linear Unit (ReLU), Leaky-ReLU, sigmoid and hyperbolic-tangent functions.

shown in figure 2.2.

Activations:

$$\text{relu}(x) = \max(0,\, x) \tag{2.16}$$

$$\text{lrelu}(x) = \begin{cases} -\alpha x & \text{if} \quad x < 0,\ \alpha \in \mathbb{R} \\ x & \text{if} \quad x \geq 0 \end{cases} \tag{2.17}$$

$$\text{sigmoid}(x) = \frac{1}{(1 + e^{-x})} \tag{2.18}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{(e^{x} + e^{-x})} = 2 \cdot \text{sigmoid}(2x) - 1 \tag{2.19}$$

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_k e^{x_k}},\ \mathbf{x} = \{x_i, ..., x_k, ...x_K\} \tag{2.20}$$

In regression problems, usually the activation function σ_k in the final layer k is omitted, making the output from the network only Z_k. For a layer j where an activation is applied, the output H_j becomes:

$$H_j = \sigma_j(Z_i) \tag{2.21}$$

$$\begin{bmatrix} h_1^j & h_2^j & h_3^j \end{bmatrix} = \left(\begin{bmatrix} \sigma_j(z_1^j) & \sigma_j(z_2^j) & \sigma_j(z_3^j) \end{bmatrix} \right) \tag{2.22}$$

The predictions $\hat{y}_i$ are made on each of the training inputs $x_i \in X$. These predictions are the outputs H_k from the last layer k of the network for each sample. The output, H_k is a function of many operations throughout the network as shown in equation (2.25).

$$H_k = \sigma_k(Z_k) = \sigma_k(H_j W_{jk} + B_k) \tag{2.23}$$

$$H_k = \sigma_k(\sigma_j(Z_j) W_{jk} + B_k) \tag{2.24}$$

$$H_k = \sigma_k(\sigma_j(x_i W_{ij} + B_j) W_{jk} + B_k) \tag{2.25}$$

In order to train the network, some error or 'loss' calculation $\mathcal{L}$ is performed between the predicted

$\hat{y}_i$ and true y_i labels for each training sample x_i. Commonly the MSE is used as shown in equation (2.26).

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.26}$$

The binary cross-entropy loss shown in equation (2.27) is also used in this work. Where N is the number of classes; y_c is 1 if the predicted class is the correct class (0 otherwise) and p_c is the predicted probability that the prediction is of the correct class.

$$\mathcal{L} = -\sum_{c=1}^{N} y_c \log(p_c) \tag{2.27}$$

The aim of training a DNN is to accurately predict y. This is done by minimize the loss $\mathcal{L}$. This can be achieved with the gradient descent method where taking a step η in the direction opposing the loss gradient will lead to lower values of the cost function. Thus, weights W in the network are updated such that:

Gradient Descent: $$W = W - \eta\frac{\partial \mathcal{L}}{\partial W} \tag{2.28}$$

The update to the weights in equation (2.28) is done gradually over many iterations and is controlled by the small Learning Rate (LR) η. If the LR is too large, the updates to the weights may cause the value of the loss function $\mathcal{L}$ to move too far in the next iteration causing it to increase, rather than decrease. Conversely, if the LR is too small, the optimizer may become 'stuck' in some local minimum.

Calculating the partial derivative of $\mathcal{L}$ with respect to the final layer weights W_{jk} is easily accomplished, however finding the gradient of $\mathcal{L}$ with respect to the first layer's weights W_{ij} is more challenging. This is where back-propagation, or implementing the chain rule, is employed. The gradient of $\mathcal{L}$ with respect to the weights W_{ij} is shown in Equation (2.29) for the 3-layer network.

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \frac{\partial \mathcal{L}}{\partial W_{jk}} \cdot \frac{\partial W_{jk}}{\partial W_{ij}} \tag{2.29}$$

In Fully-connected Neural Networks (FCNNs) the number of weights in the model is large and thus the computation of the gradient can rapidly become more intensive due to the greater number of operations being performed. Advances in processing power and computer memory in recent years has seen larger and more complex DNNs being employed on bigger and more complex input data.

In CV, images are used as the input to DNNs. Such networks achieve excellent performance on handwritten digit classification [97] even with only 2 or 3 layers. The images have to be vectorized

into a single record where each pixel intensity value is then passed to separate nodes in the input layer of the network. The images tend to be very small with dimensions in the order of 10s of pixels.

The large number of weights in FCNNs offers a redundancy in the model which makes them prone to overfitting to the training data. FCNNs do not scale well to modern color images of much larger sizes. Additionally, the FCNNs outlined above disregard spatial information in the images.

2.2.3.1 Convolutional Neural Networks.

When the input to a model has some spatial structure, *e.g.* an image, a model may be able to leverage the spatial relationship between features in order to improve its performance. This is particularly important when the subject in the image may appear at different spatial locations in different images. Translation invariance may be useful in some instances. This is not the case with MLPs: the nodes located where the subject may appear will be updated to better recognise it, but as the subject moves, different nodes are updated each time.

With Convolutional Neural Networks (CNNs), sets of filters or 'kernels' are learnt such they each become better at extracting particular kinds of features in the dataset. Each kernel is a collection of nodes, conventionally 3×3 or 5×5 nodes each. Rather than having all nodes contribute to the output at each position, the kernel is 'moved' across the input image in a sliding-window fashion from top-left to bottom-right. During the sliding window process a set of nodes in the input are covered by the kernel. The convolution, *i.e.* element-wise product and summation, of the kernel with these nodes is taken. The result is stored in the position corresponding to the center of the kernel in the output. The process repeats at each sliding window location. Thus only information from a small region surrounding a point is responsible for the resulting output.

When the convolution has been done at each point in the sliding window, the result is passed through an activation function to produce a new 'image' referred to as a 'feature-map'. The feature-map has high intensity values where the input image contains features that match the kernel that was used. For example, a kernel used to detect a straight edge, such as the Sobel filter discussed in section 2.3.2 on page 56, will create a feature map which shows all of the corresponding edges in the original image. The convolution process is shown in figure 2.3 on the following page. Many kernels can be applied at each layer of the CNN which each produce their own feature maps.

In this work, the $[B$(atch), H(eight), W(idth), D(epth), C(hannels)] convention is used to described the shape of layer inputs, filters and outputs. Thus a single grayscale, single-channel 2-dimensional image of 64×64 pixels is denoted $[1, 64, 64, 1, 1]$. From here, the batch-size will be omitted and a 2D image is assumed, so the notation becomes $[64, 64, 1]$ for a 2D grayscale image of 64×64 pixels. Similarly, any filters will be written the same way, *e.g.* $[5, 5, 1]$ implies a 5×5 node kernel with a single channel. The channel is important when considering RGB color images where kernels can either be applied on a per-channel basis, or consider all 3 channels simultane-

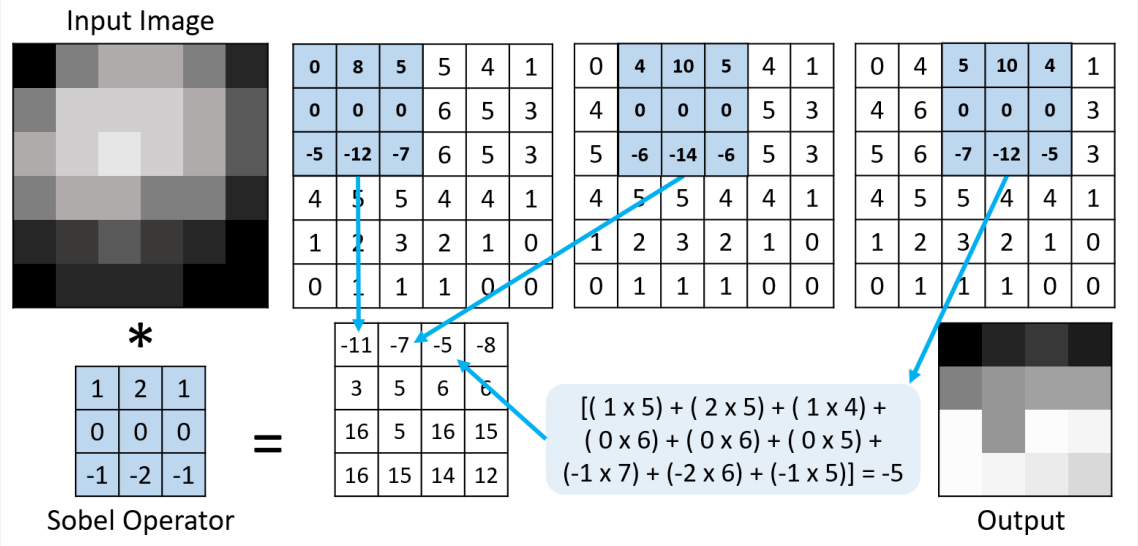


Figure 2.3: Example of the Convolution Operation. (left) The kernel (bottom) to be convolved with the image (top). Example kernel is the Sobel horizontal edge filter. (right) The convolution process that creates the output (bottom- right). Shaded region shows the multiplication when the filter is overlaid on the image.

ously. Examples of 2D and 3D images with different numbers of channels are shown in figure 2.4 on the next page.

Rather than updating the weights connecting each and every node between layers, a CNN updates the values in its filters on each iteration. Learning small kernels significantly reduces the number of weights that must be learned at each layer and allows the model to react to features present in the image regardless of their spatial location. Each filter has its own bias term.

When a kernel is convolved with an image, it will only operate where the entire filter fits onto the image. All of the values within the filter area contribute to the output which is placed at the centre of the filter. There is no position that the filter can take at the edges such that all edge positions gain a corresponding output. Therefore the output image is smaller than the input image by $(k-1)/2$ pixels in each dimension. To avoid this, padding is added to the input image before convolution. Commonly a border of zero-valued pixels is added as demonstrated in figure 2.5 on the following page. This method of padding is referred to as 'same' padding as it produces an output which is the same size as the input. The 'full' padding method ensures that every pixel in the input is visited the same number of times and produces an output larger than the input.

The size of the output image from a convolution also depends on the 'stride'. Stride indicates the number of pixels that the kernel moves each time during the sliding widow process. A stride of a single pixel means that every input pixel in the original image gains a corresponding output in the resulting image, with the caveat of the padding issue already mentioned. A stride of two pixels instructs the filter to skip a pixel at every movement of the sliding window process, giving an output

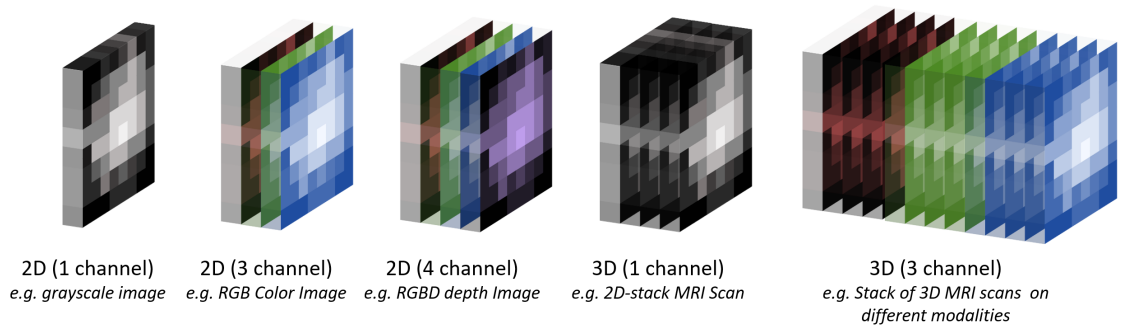


Figure 2.4: Visualization of 2D and 3D Images with Varying Numbers of Channels. 2D and 3D images can have a single channel or multiple channels. (left-to-right) Conventional grayscale images are 2D with single channel; colour images typically have 3 channels; depth information can be added as a 4th channel; medical images can be 3D with a single channel; but may also contain information from other modalities as additional channels.

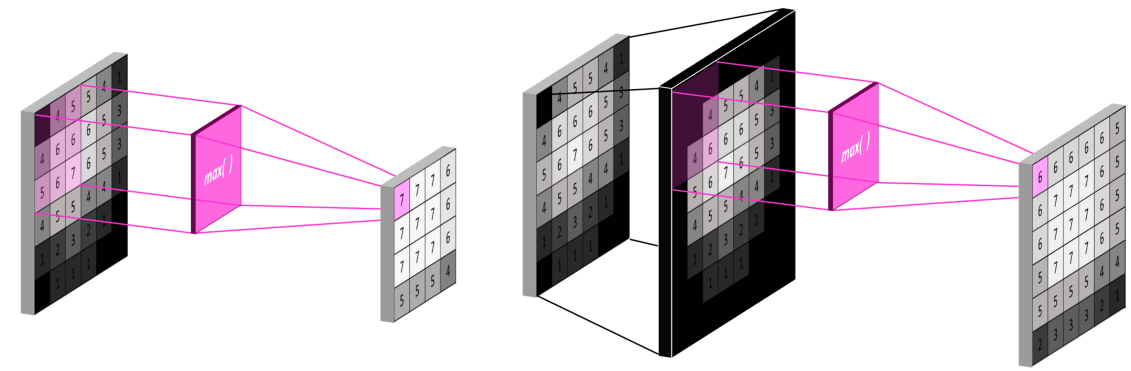


Figure 2.5: Visualization of the Effect of Padding on Convolution. A 'max-pooling' kernel is applied to the input image. (left) Without padding, the kernel moves across the image node-by-node top-left to bottom-right, but only operates where the entire filter is present. The result is an image with dimensions smaller than the input. (right) Zero-padding the input allows the filter to process the edges of the image and the output is of equal size to the input.

for every other pixel in the input image. The output image for a convolution with a stride of two is half of the original image size. The relationship between width W, kernel size K, padding P, and stride S is shown in equation (2.30).

$$\mathrm{O} = \frac{(\mathrm{W} - \mathrm{K} + 2\mathrm{P})}{\mathrm{S}} + 1 \tag{2.30}$$

CNNs which consist only of convolutional layers (conv) are referred to as Fully-convolutional Networks (FCNs). It is common in literature and online to see both Fully-connected Neural Networks and Fully-convolutional Networks referred to as FCNs; from here the acronym is used to mean Fully-convolutional Network. In a FCN, the stride parameter is often used to reduce the size of the input from one layer to the next until the final layer becomes the desired size for classification or

regression purposes.

Other 'layers' can be applied after convolutions in order to improve the performance of the CNN.

A Pooling Layers.

Some layers of a CNN may use predefined kernels rather than learning new ones. A 'pooling' layer produces an output which is smaller than the input by applying 'max' or 'average' filters. A max-pooling layer takes the largest value within the kernel at each position and uses this as output. Average-pooling layers take the average value instead. Other functions can be used, but max-pooling has been seen to work well in practice [144]. These layers can identify areas in the previous layer's output with the highest activations, *i.e.* where features corresponding to the learned kernels are present. An example of max pooling is shown in figure 2.5 on the preceding page.

Pooling applies downsampling to the image which reduces the number of parameters, reduces memory consumption and reduces the computation performed by the network. As such, it also reduces overfitting. Pooling can be used to change the size of the output instead of changing the stride parameter.

B Dropout Layer.

The problem identified with Fully-connected Neural Networks was the high capacity of the network which leads to overfitting on the training data. To combat this, a 'dropout' layer [72] can be employed which randomly sets some of the node activations to zero on each training iteration. The nodes are 'dropped' with a particular probability which is passed as a parameter to the layer. The process ensures that the network cannot become overly reliant on particular nodes. Dropout is used during training to stop updates to some nodes, but it is skipped at test time.

C Batch Normalization Layer.

The CNN does not see the entirety of the training set $\mathbf{x} = \{x_1, ..., x_i, ...x_N\}$ at any one time. Instead, the data are fed to the network in batches $\mathbf{b} = \{b_1, ..., b_j, ...b_M\}$, $b_j \subset \mathbf{x}$. Though they may be drawn from the same distribution, each batch may represent some extreme of the training distribution causing abnormally high activations for some nodes. As such, the network performs much larger weight updates than are necessary on each iteration.

To combat this, Batch-normalization (BN) or 'batchnorm' [80] is applied to the output of a layer. The mean μ_b and standard deviation σ_b of each batch is calculated as shown in equation (2.31) on the next page and equation (2.32) on the following page respectively. The outputs are then normalized to have zero mean and unit standard deviation before being passed to the next layer, as in equation (2.33) on the next page.

$$\mu_b = \frac{1}{N}\sum_{i=1}^{N} b_i \tag{2.31}$$

$$\sigma_b = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(b_i - \mu_b)^2} \tag{2.32}$$

Whilst this avoids a shift of the input distribution to each layer over time - called internal covariate shift - the process means that most values in the input fall around zero. Many of the non-linear activation functions described above have linear behaviour around zero, making these values less useful to the network. So, the normalized data are re-scaled by a standard-deviation γ and shifted by a mean value β which are learned during training. This is shown in equation (2.34). When the input to each layer is normalized, the activations become a little noisier providing a slight regularizing effect on the network.

$$\hat{b_i} = \frac{b_i - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} \tag{2.33}$$

$$\text{BN}(b_i) = \gamma\hat{b_i} + \beta \tag{2.34}$$

D Fully-connected Layer.

Fully-connected (FC) or 'dense' layers operate as in the MLP. They take a flattened vector input and a value defining the number of nodes in the output. Each node in the input is connected to each output node. One or more FC layers are usually found at the end of the network connecting the result of the convolutions to the output layer.

A form of dense layer can also be implemented by applying a 'global' pooling layer: a pooling layer which takes the maximum or average value of an entire feature-map reducing the $[H,\, W,\, C]$ output of the previous layer to a vector of shape $[1,\, C]$. A regular FC layer is applied if C is not the required number of output nodes.

E Activation Layer.

It is useful to consider the activation function applied to the output of each layer as a layer in its own right. The flow of data through a simple feed-forward CNN may then easily be written, *e.g.* `input > conv > ReLU > BN > conv > ReLU > BN > FC > dropout > output`. More complex structures may require additional diagrams or information about which layers' outputs connect to which layers' inputs.

F Output Layer.

The size, and activation, of the output layer in a CNN determines the prediction task for which is it trained. A global classification of an input into one of K classes requires K nodes in the output layer, or even a single node for binary classification, *i.e.* $K = 2$. The `softmax` activation is likely applied to give a normalized likelihood of the input belonging to each class, or the `sigmoid` activation in the case of binary classification. In global regression problems where the aim is to predict the continuous values of the GT labels, no activation is applied. Local classification problems try to assign a label to each pixel in the image, known as 'segmentation'. The label could be a deterministic, hard, single class or it could be a distribution indicating the likelihood of the pixel belonging to each class. Regression problems may also be performed at the local level, *e.g.* predicting the magnitude and direction of flow at each pixel, or predicting an overall deformation field.

2.2.3.2 Generative Adversarial Networks.

Beyond predicting classes or continuous values, this work also uses DL to generate images. Generative Adversarial Networks (GANs) are DL frameworks whose outputs are randomly generated examples that resemble the training data, but that do not actually exist. Famously, GANs have been used to produce 'deep fakes': high-quality images and videos of celebrities and world leaders which are deceivingly realistic. DL can also be used to detect such fake examples [64].

Many flavours of GAN exist. Traditional models generate images from a vector of random noise sampled from, for example, the normal distribution. Other frameworks aim to alter the appearance of an image, for example from night to day, by performing 'style transfer' [53]. The latter takes an image as input and outputs an image of the same dimensions. These GANs are referred to as Image-to-Image (I2I) translation networks which are employed in chapter 5 on page 136 of this work.

2.2.4 Image-to-Image Translation.

In Image-to-Image (I2I) translation networks, the input and output are both images. The network usually consist of downsampling and upsampling operations like those found in U-Net [137]. The idea is for the network to learn some higher level latent representation of the image space by performing convolutions on the data with learned filters. Once 'encoded', the latent representation can be decoded via transposed or 'up' convolutions back to image dimensions in order to produce a final image.

Several I2I translation networks have been proposed. Pix2pix [81] uses a conditional GAN which learns the mapping from the input image to an output image. In order to train this network, there is a need for corresponding pairs of images showing the network what the expected output is for a particular input. It is a *supervised* I2I translation framework. It is not always the case that a

corresponding output image exists. Consider the medical imaging scenario where one may wish to generate a CT scan from an MRI image. To train Pix2pix, the same subject would need to have one of each type of scan, and there would need to be a lot of these pairs from different subjects in order to train the network.

CycleGAN [184] alleviates this problem by performing 'unpaired' I2I translation. This is achieved by introducing a second generator which learns the mapping from *output to input* at the same time as the original generator learns the mapping from input to output. The CycleGAN adds a 'cycle consistency' term to the loss function during training, telling the network that the output from the second generator should very closely resemble the original input.

A extension to the CycleGAN is employed in this work for I2I translation between brain MRI acquired at different sites and in different scanners. The work is presented in chapter 5. Here, the general CycleGAN framework is outlined as a basis for this work.

2.2.4.1 Training Image-to-Image Translation Networks.

Any generative model requires both a discriminator D and at least one generator G. The discriminator aims to identify whether a given input X is from the *real* training dataset distribution or the *fake* generated distribution. The output from the discriminator is $D(X) \in [0.0, 1.0]$ where scores closer to 1.0 indicate that the discriminator believes that the input is *real* and scores closer to 0.0 indicate that the the discriminator believes that the input is *fake*. In training, the ℓ_2 loss is minimized between $D(X)$ and 0 for truly fake and 1 for truly real samples.

In the I2I translation case, 'real' and 'fake' correspond to the two image domains - the source domain S and target domain T, *e.g.* night and day, MRI and CT, or scanner 1 and scanner 2. The discriminator loss function is shown in equation (2.35). With each iteration, the discriminator should become better at differentiating between the two domains making it more difficult for the generator to produce convincing fakes.

Discriminator Loss
$$\mathcal{L}_{dis} = \frac{1}{2}\left(\ell_2(D(S), 0) + \ell_2(D(T), 1)\right) \tag{2.35}$$

In the CycleGAN, there are two generators: G_{ST} transfers the source domain image S to the target domain T which produces transformed image $S2T$; and G_{TS} transfers T to S to produce transformed image $T2S$. To recover the original images, $S2T$ can be passed through G_{TS} to get image $G_{TS}(S2T) = S2T2S$ and $T2S$ can be passed through G_{ST} to get $G_{ST}(T2S) = T2S2T$.

There are two conditions that the generators try to meet: (1) $S2T$ and $T2S$ must fool the discriminator, and (2) $S2T2S$ and $T2S2T$ must be consistent with S and T respectively. Condition 1 is enforce by the adversarial loss $\mathcal{L}_{adv}$ in equation (2.36) on the next page and condition 2 by the cycle consistency loss $\mathcal{L}_{cyc}$ in equation (2.38) on the following page. $\mathcal{L}_{adv}$ is applied to discriminator outputs whilst $\mathcal{L}_{cyc}$ acts directly on the image data. The total generator loss is the weighted

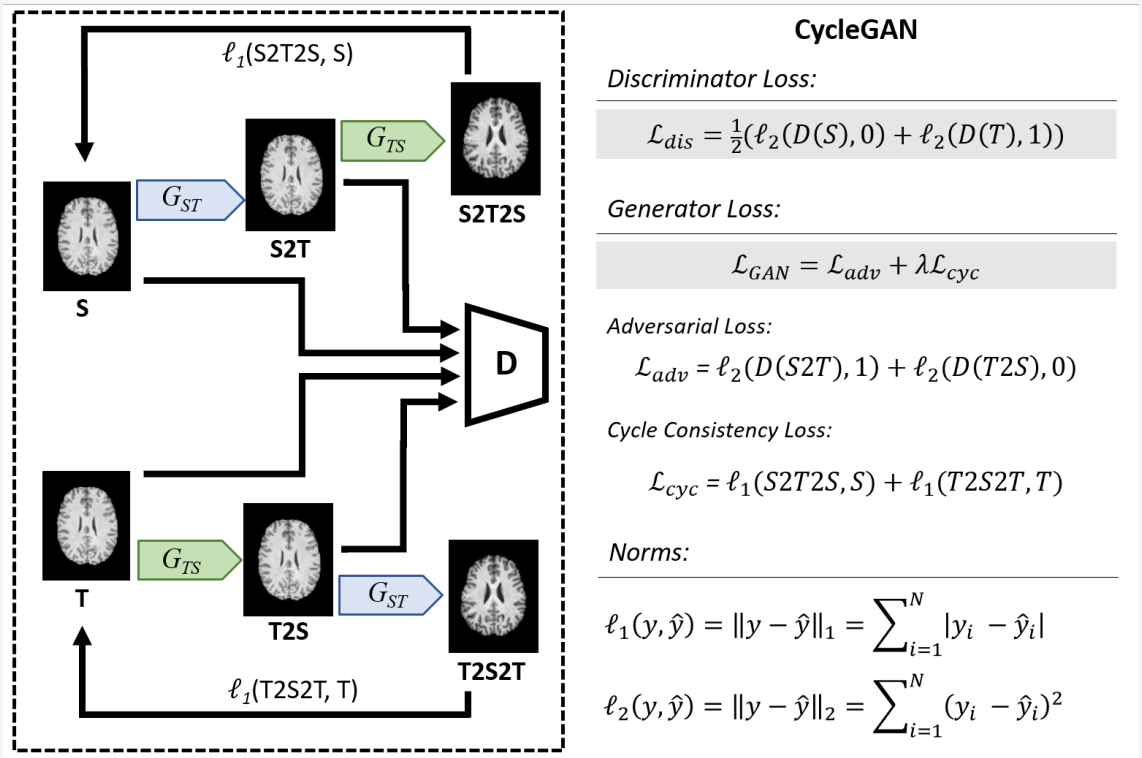


Figure 2.6: Visualization of CycleGAN and Associated Losses. (left) the CycleGAN framework showing the discriminator D and two generators G_{ST} and G_{TS}. $S2T = G_{ST}(S)$, $T2S = G_{TS}(T)$, $S2T2S = G_{TS}(S2T)$ and $T2S2T = G_{ST}(T2S)$. (right) the associated losses.

sum of these terms in equation (2.40) where λ determines the importance of $\mathcal{L}_{cyc}$. The weighting term $\lambda = 10$ in the original paper [184]. The CycleGAN and associated losses are visualized in figure 2.6.

Adversarial Loss
$$\mathcal{L}_{adv} = \ell_2(D(S2T), 1) + \ell_2(D(T2S), 0) \tag{2.36}$$
$$= \ell_2(D(G_{ST}(S)), 1) + \ell_2(D(G_{TS}(T)), 0) \tag{2.37}$$

Cycle Consistency Loss
$$\mathcal{L}_{cyc} = (\ell_1(S2T2S, S) + \ell_1(T2S2T, T)) \tag{2.38}$$
$$= \ell_1(G_{TS}(G_{ST}(S)), S) + \ell_1(G_{ST}(G_{TS}(T)), T) \tag{2.39}$$

Generator Loss
$$\mathcal{L}_{gan} = \mathcal{L}_{adv} + \lambda\mathcal{L}_{cyc} \tag{2.40}$$

2.2.5 Model Evaluation.

During training, the loss function serves as a proxy for how well the model fits the data at the current iteration. It may also calculate a series of metrics at each step, such as accuracy, which may be more intuitive to the developer than the value of some complex loss functions.

Once a predictive model is trained, it is important to assess its performance before deploying it

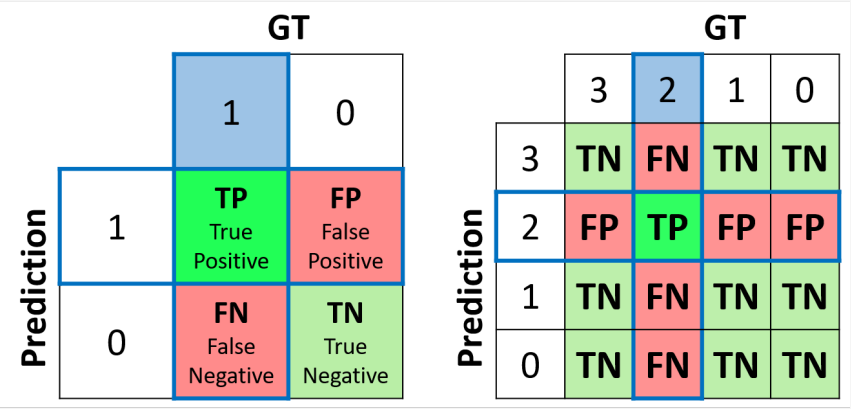


Figure 2.7: Confusion Matrices. (Left) A confusion matrix summarising the prediction results of a binary classification model compared against Ground Truth (GT) labels. The four descriptors are (1) True Positive (TP) - the prediction and GT are both positive; (2) FP - the prediction is negative and GT is positive; (3) FN - the prediction is positive and GT is negative; and (4) TN - both prediction and GT and negative. (right) Extension to the multi-class case of 4 classes shown for the case where GT label = 2.

in the real-world. It must accurately predict the training data whilst 'generalizing' to new, unseen examples. Several metrics are commonly used to measure the performance of a ML model, many of which stem from the fundamental idea of a 'confusion matrix'.

2.2.5.1 Confusion Matrix.

Let 0 and 1 be two classes in a binary classification task where 0 and 1 represent negative and positive outcomes respectively. When the GT label of a sample is 1, the task model $\mathcal{T}$ will either predict class 1 (True Positive (TP)) or class 0 (False Negative (FN)). Conversely, if the GT label is 0, the model may also predict class 0 (True Negative (TN)) or class 1 (False Positive (FP)). The outcomes can be summarised in a confusion matrix as in figure 2.7. When the model output for each class is a floating value, some threshold is set in order to determine whether the prediction is positive or negative. The idea can be extended to the multi-class classification scenario as shown on the right of figure 2.7.

From the confusion matrix, a number of metrics can be defined.

1 **Accuracy.** The number of correct predictions out of all predictions made, presented as a dimensionless quantity or a percentage. A model that predicts perfectly has an accuracy of 1.0 or 100%. Accuracy can be misleading in cases where there is an imbalance in the number of samples of each class. A dataset where 1 sample is positive and 99 are negative may train a model that predicts the negative class for any input, thus achieving 99% accuracy. Other

measures are more informative about the distribution of predictions.

Accuracy.
$$\text{Acc.} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.41}$$

2 **Recall/Sensitivity/True Positive Rate.** The True Positive Rate (TPR) measures the proportion of actual positives that are correctly identified as being positive.

Recall.
$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.42}$$

3 **Specificity/True Negative Rate.** The True Negative Rate (TNR) measures the proportion of actual negatives that are correctly identified as negative.

Specificity.
$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.43}$$

4 **Precision/Positive Predictive Value.** Out of the number of times that the positive class was predicted, Positive Predictive Value (PPV) is how many times was it predicted correctly.

Precision.
$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.44}$$

5 **Fall-out/False Positive Rate.** The proportion of all negative GT samples that yield positive predictions. Equal to $(1 - \text{specificity})$

Fall-out.
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.45}$$

6 **F1-score.** The harmonic mean of precision and recall. A model that predicts perfectly has an F1-score of 1.0. The same formulation defines the Dice Similarity Coefficient (DSC), also referred to as Sørensen–Dice coefficient or Dice score, which is used extensively in this work.

F1-score.
$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.46}$$

2.2.5.2 Logarithmic Loss.

Models may be designed to give soft labels or probabilities of a sample belonging to each of the available classes. The logarithmic or 'log loss' is defined in equation (2.47) on the next page for N samples and M classes where y_{ij} indicates whether sample i belong to class j and p_{ij} indicates the probability of sample i belonging to class j. The log loss penalizes false classifications. A

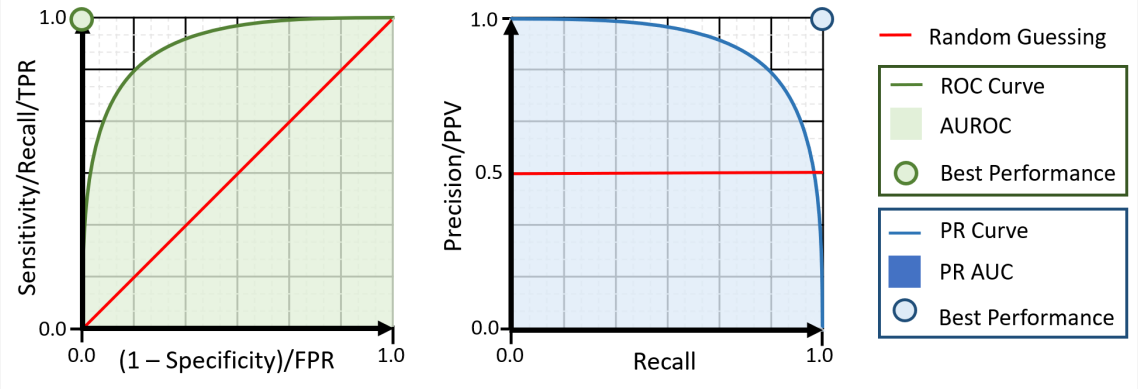


Figure 2.8: Receiver Operating Characteristic and Precision-Recall Curves. (Left) The Receiver Operating Characteristic (ROC) and (right) Precision-Recall (PR) curves showing AUC (shaded). Best performance points are at $(0,\ 1)$ and $(1,\ 1)$ for ROC and PR respectively. A classifier with the performance of random guessing is shown in the case of balanced classes. Note that the PR curve is a mirror image of the ROC curve.

value closer to 0 indicates higher accuracy of the predictions.

Logarithmic Loss.

$$\text{Log Loss} = \frac{-1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M} y_{ij}\log p_{ij} \tag{2.47}$$

2.2.5.3 Area Under Curves.

A Receiver Operating Characteristic (ROC) is the plot of TPR against False Positive Rate (FPR). Area Under the Receiver Operating Characteristic (AUROC) or Area Under Curve (AUC) refers to the area below the characteristic curve in this plot. The TPR and FPR are calculated using a series of thresholds on the model output and placed onto the graph. The area below the resulting curve is calculated. TPR $\in [0,\ 1]$ and FPR $\in [0,\ 1]$ thus AUC $\in [0,\ 1]$. A random or 'uninformed' classifier gives AUC $= 0.5$. The AUC is equal to the probability that the predictor ranks a randomly chosen positive sample higher than a randomly chosen negative one [48].

Points on the curve falling above the graph diagonal indicate better than random performance of the predictor model and those on the diagonal indicate random performance. The best possible predictor would yield a point at the upper-left corner of the plot at TPR $= 1$ and FPR $= 0$.

AUROC should not be confused with the area under a Precision-Recall (PR) curve where a perfect predictor has the point $(1, 1)$ and a random classifier is identified by a horizontal line with precision proportional to the number of positive samples. The line sits at precision $= 0.5$ for a random binary classifier on a balanced dataset. PR AUC may be a more effective metric for classification models where the data are imbalanced [15]. Examples of ROC and PR curves are shown in figure 2.8 with their associated properties .

Section 2.3, puts the ML and DL methods detailed in this section into practice. Methods for image segmentation are described.

2.3 Image Segmentation.

2.3.1 Introduction.

Segmentation is the process of partitioning an image such that image elements, pixels (2D) or voxels (3D), are separated into disjoint regions by some boundary. Semantic segmentation is the process of partitioning an image by assigning a label to each image element. This is image classification at the image-element level. Elements in different regions, separated by boundaries, may share the same class label, *e.g.* background.

An image segmentation may simply separate the foreground and background components of an image. The segmentation may comprise multiple objects, each segmented and assigned their own class label. The label assignment may be deterministic, *i.e.* hard labels, or indicate the likelihood of an image element belonging to each class in the segmentation, *i.e.* soft labels.

Segmentation methods vary from simple global operations on pixel intensities (section 2.3.2); to statistical models of shape and appearance (section 2.3.3); and to state-of-the-art Deep Learning (DL)-based approaches (section 2.3.4).

Most algorithms discussed in this section can be applied to 2D or 3D images, thus the words 'pixel' and 'voxel' may be used interchangeably.

2.3.2 Traditional Image Segmentation Methods.

Those segmentation methods that do not rely on Machine Learning (ML) or label-propagation approaches are, in this work, categorised as 'traditional' segmentation techniques. Such approaches can be grouped into 'region' and 'edge' methods, utilizing similarities and discontinuities in the image respectively.

2.3.2.1 Intensity-based Segmentation Methods.

The simplest form of image segmentation performs a thresholding operation on pixel intensities. Variations of image thresholding also exists.

Thresholding.

$$f(x) = \begin{cases} 0 & \text{if} \quad x < \text{threshold} \\ 1 & \text{if} \quad x \geq \text{threshold} \end{cases} \tag{2.48}$$

For an 8-bit image with a single channel (grayscale), the intensity value of a pixel is in the range $[0, 255]$. To 'threshold' an image, a threshold value is chosen within the pixel intensity range. Any pixel with an intensity greater than or equal to the threshold is given a value of 1, and 0 otherwise. These pixels are considered to belong to the positive class and have been 'segmented' from the

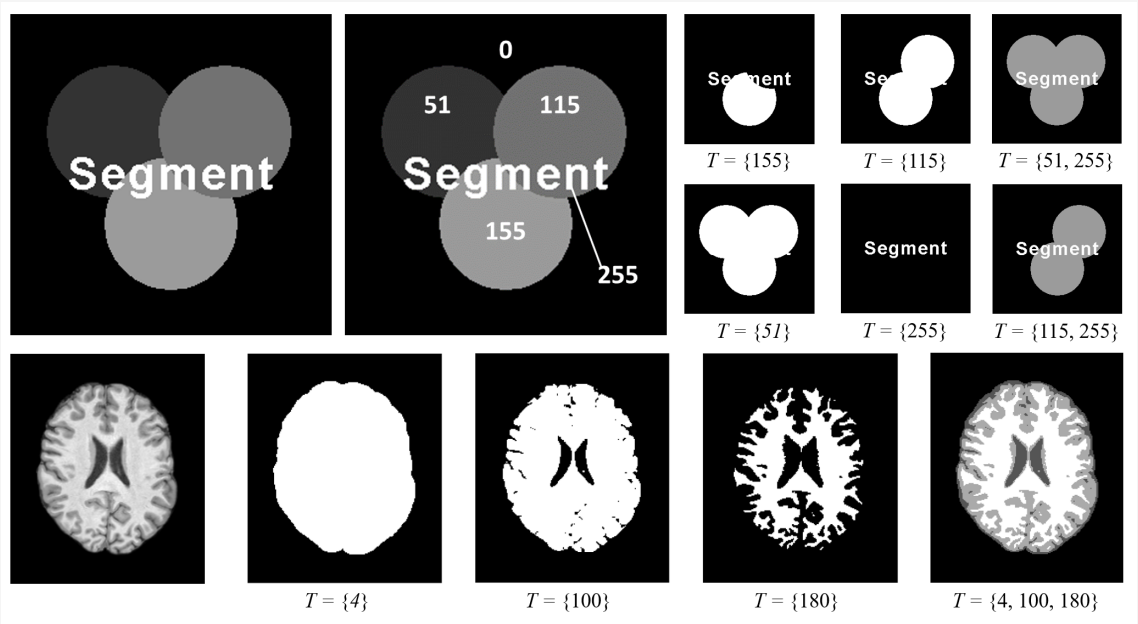


Figure 2.9: Example of Image Segmentation by Thresholding. (top) Toy example showing (left) the image to be segmented. Pixel intensity values indicated on the second image. (right) Results of segmentation using the tresholding method. Thresholds T applied to the image are indicated below the segmentation. Single threshold applied in the first two columns. 2 thresholds applied in the third column. (Bottom) Thresholding applied to brain Magnetic Resonance Imaging (MRI) (left). Global thresholds need to be selected carefully to avoid mis-classifying the ventricles as background. Images were generated with the code in listing 6.1 on page 205

image. The thresholding function $f(x)$ is shown in equation (2.48) on the previous page. Of course, more threshold values can be provided in order to segment an image into multiple classes. In the case of N threshold values $\mathcal{T} = \{t_0, t_1, ..., t_N\}$ the tresholding function $f(x)$ is given in equation (2.49). The resulting labelmap, *i.e.* segmentation, in any case is the same size as the original image.

$$f(x) = \begin{cases} 0 & \text{if} \quad x < t_0 \\ i & \text{if} \quad t_{i-1} \leq x < t_i \quad \text{and} \quad 0 < i < N \\ N & \text{if} \quad x > t_N \end{cases} \tag{2.49}$$

When thresholding is performed over the entire image at once, it is referred to as 'global' tresholding. However, there may be instances where this leads to undesirable segmentation results. As an example, consider an image where parts of the background contain pixels of equal or higher intensity than the object of interest in the foreground. Global thresholding will not only segment the required region, but also those parts of the background of equal or higher intensity creating unwelcome false positives. To mitigate this problem, 'local' or 'adaptive' thresholding can be performed.

Rather than set the thresholds manually, an optimized set of thresholds can be found by maximizing the variance between classes, equivalent to minimizing intra-class intensity variance. This is known as Otsu's method [118]. Other methods use the histogram shape, clustering, or entropy calculations as discussed in [147].

With local thresholding [14], a *different* threshold is applied to each pixel based on characteristics of its neighbours. This can be achieved with a sliding window method. Again, multiple thresholds can also be used. The advantage here is that local thresholding can segment an image into multiple classes based on regionally sensitive threshold values. Alternatively, adaptive thresholding applies a threshold to the intensity value of each pixel which is a function of its spatial location in the image. This method may also take into account additional information held at each pixel when each is vector of data.

In medical image analysis, the images often contain a large number of anatomical features with fine structure. The voxels have a large range of intensity values and the contrast between different structures may not be very high. Disparate volumes may have similar intensity statistics. Thus, thresholding may not provide the best segmentation result in many cases [49]. Tresholding techniques are sensitive to noise which, for some imaging modalities, *e.g.* ultrasound, means they may not work well.

Examples of segmentation by thresholding are shown in figure 2.9 on the previous page.

2.3.2.2 Region-based Segmentation Methods.

Methods in this category generally perform some 'similarity' comparison between pixels of the image with most focused on pixel intensity. Pixels are assigned to classes based upon their similarity to each other or to some chosen reference point. The methods work either in the image space or some other feature space, *e.g.* on histograms of intensities.

A threshold is also needed in the region-growing method. In addition, it requires a number of 'seed' points to be initialized by choosing some pixels. This may be done automatically, randomly or interactively by a human. The number of seeds indicates the number of salient regions, or segments, that should be constructed. The pixels neighbouring the seeds are assessed: if some condition is true, the neighbouring pixel is subsumed into the region occupied by the seed, otherwise it is ignored. Conditions may be simple metrics such as the absolute value of the difference between the seed and candidate pixel's intensities. Some threshold on this difference is set. At each step, pixels neighbouring the growing region are assessed and added to the region's mass given the condition is met. Another approach is to compare the unassigned neighbours with the outermost pixels of the region which may improve the method's sensitivity to noise.

Region-growing can also be done by specifying only a single seed: neighbouring pixels are compared and added to the initial region based on some criteria, but a new region is created if the

difference is too much. This 'unseeded' region-growing technique is useful when the number of regions is initially unknown.

The computational cost of this approach is larger than with the intensity thresholding method above. However, region-growing provides good boundary information and produces disjoint segments in which the pixels form contiguous connected regions. Though weakly contrasting boundaries, *e.g.* between the sea and the sky, may cause the region growing method to 'leak' into unwanted areas. The growth criteria can be more complex and utilize any information available at each pixel.

Seed selection can be important in the region-growing method. Seeds can be chosen based on the image's intensity histogram or by creating a regular grid of seeds across the image.

Region-growing performs very local comparisons of the data and has no overall global view of the segmentation problem. This can make it unsuitable for problems where structure or semantics may be important. Region growing can be successful in medical image analysis when the homogeneity and shape of the growing regions is taken into account [87, 125]. A comparative study on region-growing techniques can be found in [46].

2.3.2.3 Edge-based Segmentation Methods.

In this category, segmentation methods are looking for discontinuities between semantic regions of an image. That is, they try to find boundaries between segments that best separate the image, usually based on pixel intensities. Localised changes in intensity can be found with gradient-based approaches.

A Differential Operators.

An edge in an image occurs where intensity values or textures undergo sudden local changes. The discontinuity in the values can be detected by differential operators that calculate derivatives over the regions. Edge detection can be done efficiently using the convolution of some filter with the image. The Sobel operator [153] is an example of a widely used first-order differential operator. The Sobel operator approximates the gradient of the image's intensity function. It convolves transverse and longitudinal filters with the image to identify horizontal and vertical edges respectively, and finds the magnitude and direction of the resulting vectors.

The Laplacian operator can also be used as an edge or isolated pixel detector by approximating the second-order differential of the image's intensity function. This operator is sensitive to noise and so is usually combined with an initial low-pass smoothing filter prior. Examples of these filters are shown in figure 2.10 on the following page.

B Watershed Segmentation.

The pixel intensities of an image can be viewed as a topological map with high intensities repre-

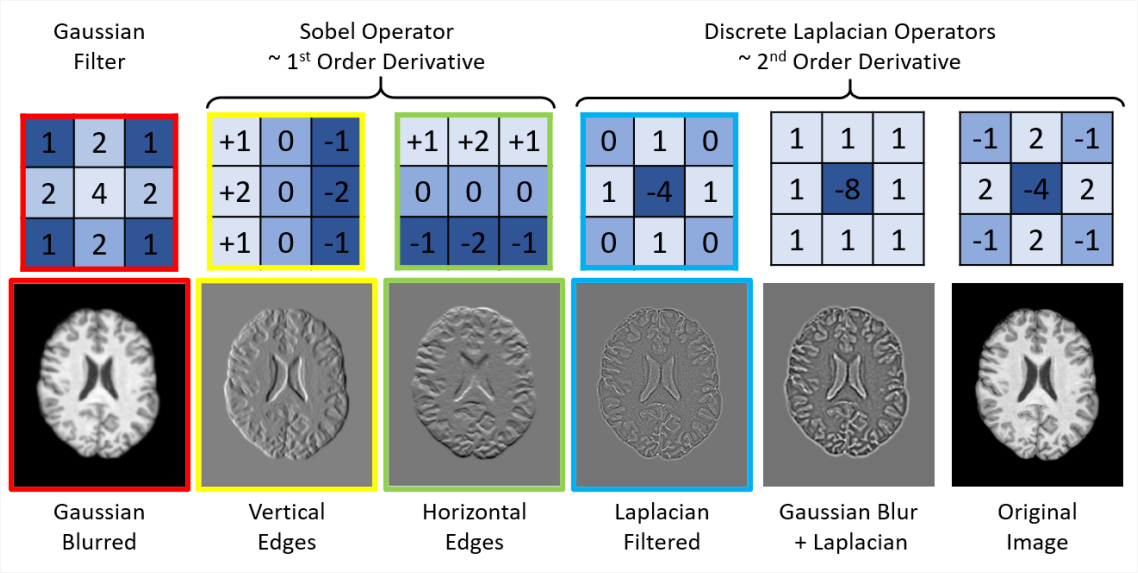


Figure 2.10: Examples of Discrete Operators used in Edge Detection. (Top) The discrete filters shown are (left-to-right) Gaussian, horizontal and vertical Sobel operators, and examples of possible Laplacian operators. (Bottom) Result of convolving the discrete filters with the original image (far right). Images were generated with the code in listing 6.3 on page 212

senting mountains and low intensities acting as valleys or basins. The 'watershed' analogy can be applied to such a topology by filling the basins, *i.e.* intensity minima, with water until they reach the peaks where 'dams' can be built. These dams act as boundaries for the image segmentation. This can be achieved by considering the gradient directions and magnitudes away from the intensity minimum until they reach the peak and change sign.

Watershed segmentation ensures that connected regions flow into one another, but the method requires pre- and post-processing. Initial reduction of salt-and-pepper noise (high-frequency components) in the image through median-filtering helps smooth more homogeneous regions whilst erosion and dilation of the resulting binary image cuts the connected components into more realistic regions.

Watershed is prone to over-segmentation, but this can be overcome by applying the algorithm over gradient-magnitude images rather than the raw image data. Such an approach was applied to facial anatomical structures [114]. The algorithm has also been readily applied to brain extraction in MRI and tumour segmentation [68].

2.3.2.4 Curve-evolution Segmentation Methods.

Curve-evolution concerns the change in a contour or boundary over time. These methods rely on Partial Differential Equations (PDEs) to represent a segmentation boundary or region by some function which changes on each iteration, or time step. The evolution of this function results in a segmentation boundary that is optimized over time by minimizing some associated cost.

A Level Sets.

Level-sets is a general numerical method for solving evolving contours or 'fronts' of a surface which change over time. It defines an implicit representation of the contour C at the zero-level set of a higher-dimensional function ϕ such that $\phi(C) = 0$. For a 2D image, the surface or 'level-set' function ϕ is a function of the image-space and time, thus $\phi(x, y, t) = 0$. The contour on the image-plane is the cross-section of ϕ at $z = t$.

The surface is moved according to a velocity term $\mathbf{v}$. For ease, the surface is moved in the direction n normal to the image-plane, reducing $\mathbf{v}$ to a speed term F. For complex contours, the speed at each pixel is commonly defined by its curvature where the pixels move perpendicular to the contour. The level-set equation is shown in equation (2.52) which describes the area enclosed by the surface which expands or contracts depending up on the sign of F. This approach works well because the level-set function is smooth and well-behaved, whereas mapping the evolution of a contour in the image-space may result in discontinuities when shapes merge or split apart.

$$\frac{d\boldsymbol{\phi}(C)}{dt} = \frac{\partial \boldsymbol{\phi}}{\partial t} \cdot \nabla \boldsymbol{\phi} + \frac{\partial \boldsymbol{\phi}}{\partial t} = 0 \tag{2.50}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial t} = \mathbf{v} = F\mathbf{n} = F\frac{\nabla \boldsymbol{\phi}}{|\nabla \boldsymbol{\phi}|} \tag{2.51}$$

Level-set Equation:

$$\frac{\partial \boldsymbol{\phi}}{\partial t} = -F\,|\nabla \boldsymbol{\phi}| \tag{2.52}$$

The method has been successfully applied in different anatomy for medical image segmentation, including extracting tumours in neuroimaging [158] and cardiac anatomy [91].

B Level-sets Modifications.

The are two notable adaptations of the level-sets algorithm which can be used for image segmentation: 'narrow-band' level-sets and the 'fast-marching method'.

Rather than updating the speed function F over the entire image, 'narrow-band' level-sets [1] modifies the algorithm to consider only the narrow region close to the surface contour. The work in [156] extended this approach and evaluated it on cell images and brain MRI.

The 'fast-marching method' [146] is also an extension of level-sets which assumes that the contour is always expanding or always contracting. Thus, at any time t the height of the surface represents the location of the contour. The method was used in [22] on intra-vascular ultrasound images.

C Active Contours (Snakes).

In image segmentation, active contours or 'snakes' [92] refers to the movement of some initial boundary annotation until it wraps along some edge in the image $\mathbf{X}$. The boundary curve is represented as a set of connected points which lie at separated pixel locations and is usually 'closed'

such that the first and last points are joined. The contour is moved by minimizing an energy function.

Two 'energies' are considered: the external energy and the internal energy. The external energy, defined in equation (2.53), measures the gradient magnitude in the image at the location of each point in the contour. The internal energy, defined in equation (2.54), indicates the distance between the point in the contour and its curvature. It takes weighting factors α and β for the 'elasticity' and 'curvature' term respectively.

$$E_{\text{external}} = \sum exp\left(-||\nabla f(\mathbf{X}_i)||\right) \tag{2.53}$$

$$E_{\text{internal}} = \sum \alpha\,||\mathbf{X}_i - \mathbf{X}_j|| + \beta\,||\mathbf{X}_{i-1} - 2\mathbf{X}_i\mathbf{X}_{i+1}|| \tag{2.54}$$

Each iteration of the algorithm aims to move the points of the contour. Different schemes can be applied, *e.g.* only allowing points to move such that the contour shrinks, or allowing points to move only a single pixel at any time.

Active contours requires some prior knowledge of the boundary shape. However, it may also fail when regions merge or develop holes. As such, the active contours method can also be considered using the level-sets formulation which allows it to handle topology changes during the curve evolution.

2.3.2.5 Graph-based Segmentation Methods.

'Graph-cuts' was first applied to computer vision in [63]. In graph methods, the image representation is changed such that the pixels become a set of vertices $V = \{v_1, v_2, ..., v_n\}$. A set of edges E connects each vertex pair $(v_i, v_j) \in E$ with some corresponding weight $w(v_i, v_j)$. The weight function is some measure of dissimilarity between the connected pixels based on intensity, motion or some other locally varying attribute. The vertices and edges create the graph $G = \{V, E\}$.

An image segmentation is just a graph partition such that each segment is a connected graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. All vertices are used but the sets are mutually exclusive. A subset of edges is a 'cut' C whose cost can be calculated as the sum of the weights on the included edges, shown in equation (2.55).

$$|C| = \sum_{e \in E'} w_e \tag{2.55}$$

A Minimum Graph Cut.

The cut that minimizes the cost in equation (2.55) is called the minimum cut [13]. Given a 'source' and 'sink' node in the graph, it separates the image into disjoint subsets: foreground F and back-

ground B such that $F \cup B = V$. However, this min cut favours small, isolated clusters. This can be mitigated by considering *normalized* cuts which take into account the volume of the sub-graphs based on edge weights.

B Clustering by Graph Eigenvalues.

The weights w on the edges between pixel pairs can be represented as an affinity matrix A. The eigenvalues and eigenvectors of this matrix can be calculated as in equation (2.56) where $\mathbf{w}_n$ represents how much the vertex j associates with cluster n. The algorithm iterates through the eigenvalues, starting with the largest, and assigns each unassigned vertex to a cluster.

$$A\mathbf{w}_n = \lambda \mathbf{w}_n \tag{2.56}$$

2.3.3 Registration-based Image Segmentation Methods.

In image registration, a source image S is warped onto a target image T such that the some similarity, or dissimilarity, measure between the two is optimized. Optimization may include a minimization of the intensity errors, or maximization of Mutual Information (MI). Image registration is introduced in detail in section 2.4, however methods of image segmentation that leverage registration are outlined here.

A Active Appearance Model Segmentation Method.

Earlier iterations of Active Appearance Models (AAMs) were developed as statistical Point Distribution Models (PDMs) [31] where model parameters were automatically learned from sets of corresponding points between images. The PDMs incorporated shape and boundary information as well as their allowed variations, but only considered image appearance near to object borders.

The same authors developed Active Shape Models (ASMs) [33] and later AAMs [30] which demonstrated model-driven segmentation methods. The AAM is a deformable model of shape and appearance which describes an image and its associated object shapes as a statistical shape appearance model. During the training phase, the mean shape and appearance of a dataset are extracted along with their principal modes of variation. The AAM consists of shape, motion and appearance models. The shape model is trained with a set of images and landmark co-ordinates, the motion model warps textures, and the appearance model extracts image features. To segment an image, the model iteratively deforms to fit to object examples in a new image.

The authors' own comparison of ASMs and AAMs [32] show three main differences: the ASM only models image texture in small areas near each landmark whilst AAMs create a model of appearance for the entire region; ASMs search along profiles normal to the boundary at the current position, whereas AAMs only sample the image at the current position; and ASMs seek to minimise the distance between model points and corresponding image points, whereas AAMs seek to minimise the difference between synthesized model images and target images.

AAMs have been shown to be robust in segmenting 3D brain MRI [7] and high-resolution foot and ankle Computed Tomography (CT) [16].

B Multi-atlas Label Propagation Segmentation Method.

An image S that has a corresponding manually annotated, or verified Ground Truth (GT) labelmap L is referred to as an atlas, $A = \{S, L\}$. In multi-atlas label propagation, N atlases $\{A_i\}_1^N$, $A_i = \{S_i, L_i\}$ are each registered to the target image T to be segmented which creates warped atlas images S'. Each labelmap $l \in \{L_i, ..., L_N\}$ associated with each atlas image $s \in \{S_i, ..., S_N\}$ is warped using the corresponding deformation field from the registration to produce warped labelmaps L'. Thus, all atlases are now aligned with T.

In weighted voting methods [5], the label at a voxel in T is determined by fusing the labels of the

corresponding voxels in the warped atlas segmentations L'. In the majority-voting label fusion method, the mode of the labels contributes the final result for each voxel where each atlas has equal weight.

Errors in the image registration process can cause S' to be misaligned. A windowing technique can be applied that takes into account the similarity of intensities between patches centered at the target voxel in T and S'. Such patch-based approaches have been successful as they allow the voting weight to spatially vary by representing local similarity between T and S' [36]. Rather than compare T to the atlas patch, [168] considers the similarity between atlas patches themselves to reduce correlated labelling errors. Often, a K-means classifier is employed to perform the voxel-wise classification based on patch similarity.

An Expectation Maximization (EM) approach can be employed for statistical label fusion rather than using the weighted voting method. For example, Simultaneous Truth and Performance Level Estimation (STAPLE) [6, 171] can be applied which estimates the label probability by combining atlas label information and a human rater performance matrix.

In medical imaging, there is a significant drawback to this multi-atlas patch-based approach: patches of different anatomy may appear with similar intensity to other tissues or even the background. This was addressed in [8] by adding gradient and contextual information to the feature vector at each pixel, and by replacing the K-means classifier with a Support Vector Machine (SVM).

C Information Bottleneck Method Segmentation Method.

The work in [10] proposes two clustering algorithms based on registration using Mutual Information (MI). In the algorithms the histogram bins of two registered images are clustered by minimizing the loss of MI between them. The clustering is driven by the preservation of shared information by extracting structures from each image which are more relevant to the other.

2.3.4 Machine Learning-based Image Segmentation.

Segmentation is an image classification problem at the pixel-level. Some ML-based classifiers have been discussed in section 2.2.2 on page 33 for classification at the global image level. These algorithms can be readily applied at the pixel-level in order to assign labels per pixel. Other algorithms are also discussed in this section.

A K-means Clustering Method.

The K-means algorithm for segmentation works in the same way as for the global image classification case in section 2.2.2.1 on page 34. It relies on some 'distance' metric calculated from the centroid of each cluster to all data samples. In image segmentation this distance relates to pixel intensity. In an RGB color image, each pixel 'intensity' is a 3-vector, one for each channel. The distance to the centroids is calculated for all pixels in the image. Each pixel is assigned to the same group or class as its closest centroid.

For large images, the resulting segmentation can often be improved by down-scaling the image before computing the clusters and then upsampling the result to assign labels to pixels at the original image size.

For medical images, the algorithm can be used to separate different tissue types, *e.g.* grey and white matter in neuroimaging data [103]. Fuzzy C-means is also useful in medical image analysis, *e.g.* in tumour segmentation [135], as it can provide some confidence metrics.

B Gaussians Mixture Model Method.

A Gaussian Mixture Model (GMM) can be applied to the histogram of intensities over pixels in the image. Peaks in the histogram can be modelled with Gaussian distributions and pixels are assigned soft labels based on the probability of belonging to a particular cluster or class. If x is the vector representing the pixel intensities, then a model P_{GMM} can be found as in equation (2.57) where $\pi_i \geq 0$ are the mixing coefficients with $\sum_{i=1}^{K} \pi_k = 1$ for the K Gaussian modes and μ_i and σi are the means and covariances of the Gaussian modes respectively. The model is fit over all pixels in the image using a maximum-likelihood approach.

$$P_{\text{GMM}}\left(x|\theta\right) = \sum_{i=1}^{K} \pi_i . \mathcal{N}\left(x|\mu_i, \sigma_i\right) \tag{2.57}$$

An example of brain tissue segmentation with GMM is shown in figure 2.11 on the following page.

C Support Vector Machine Method.

The SVM prediction models from section 2.2.2 on page 33 can be applied to image segmentation. The models learn to separate the pixels of the image into classes, such that each pixel can be labelled.

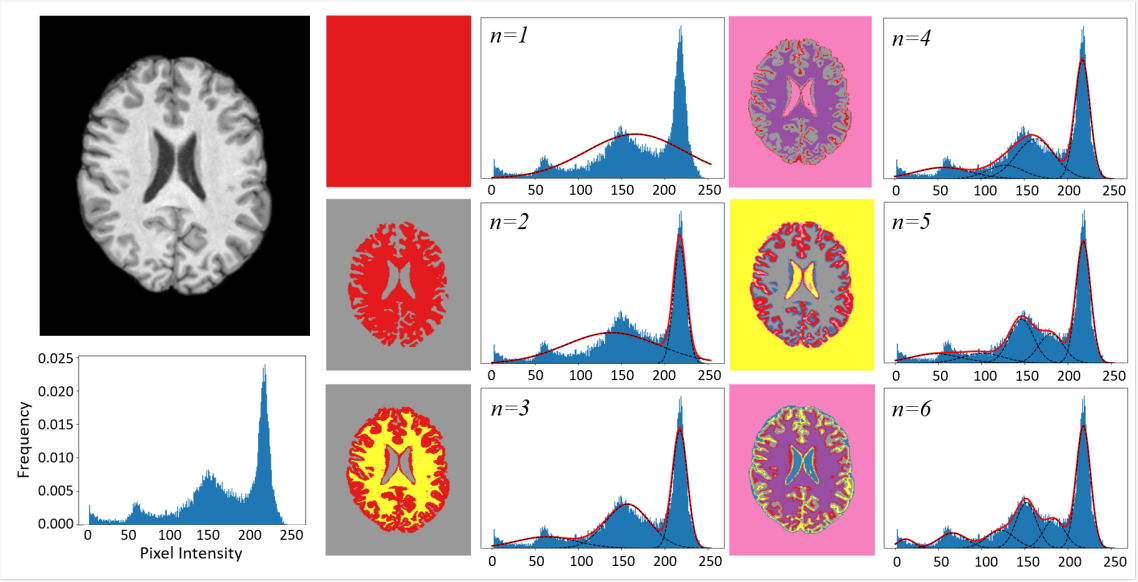


Figure 2.11: Example of Image Segmentation by Gaussian Mixture Model (GMM). (left) The image to be segmented and corresponding histogram of intensities below. (right) Results of segmentation by fitting GMM with (top-to-bottom, left then right) 1, 2, 3, 4, 5 and 6 components. Too few components incorporates ventricles of the brain into the background class, whilst too many can over segment the image. Images were generated with the code in listing 6.2 on page 209.

Medical imaging applications of SVM-based segmentations include brain MRI segmentation [67], and extracting liver tumours in CT scans [180]. The work in [26] combines the AAM approach with a SVM for prostate MRI segmentation.

D Random Forest Method.

The Random Forests (RFs) laid out in section 2.2.2 on page 33 can be used to perform image classification at the pixel-level [39], *i.e.* segmentation. During training, the image pixels are passed to the RF with their GT labelmaps. The bagging technique is used such that each tree sees a random subset, with replacement, of the training data. Feature bagging is also employed whereby each node in a tree sees only a random subset of the pixels at any time. Boosting is employed to produce many weak segmentors in succession which focus on the most misclassified pixels.

E Convolutional Neural Network Segmentor.

Convolutional Neural Network (CNN)-based segmentors usually take images as input and predict segmentations as output. GT labelmaps are used to perform supervised learning of the model. The CNNs are trained such that the difference between the predicted and GT segmentations is small.

Many CNN-based segmentors adopt the U-Net [137] architecture whereby an image is progressively downsampled to form some high-level representation of the image and then reconstructed. In medical image analysis, DeepMedic [90] is based on a multi-scale 3D CNN coupled with a 3D

Fully-connected (FC) Conditional Random Field (CRF). It has been successful in brain lesion segmentation.

Further details of DL-based algorithms for medical image analysis tasks, including segmentation, can be found in the extensive survey of [101].

The K-means and GMM approaches are segmentation methods that do not account for semantic labels. Conversely, SVM, RF and CNN-based segmentors are trained with GT labelmaps such that they are able to learn relationships between the image intensity data, and its corresponding semantic labelmap.

2.4 Image Registration.

2.4.1 Introduction.

Image registration is the process of aligning two or more images of the same scene, structure or content which have been acquired at different times, with different instruments, or from different perspectives. Images with the same alignment are more easily compared. Registration has uses in the natural image domain from stitching together individual images to create a single panoramic scene [110], to astrophotography [106] where many low-light images are combined so that fainter objects become more visible.

For medical images, the analysis may be multi-temporal and multi-modal which can lead to complex registration tasks. But registration is important for a number of cases in this domain. To compare the progress of a patient's treatment, images acquired at different time-points need to be registered. Where structural and functional images are acquired by different modalities, they need to be superimposed, *e.g.* Magnetic Resonance Imaging (MRI), Computed Tomography (CT) and Positron Emission Tomography (PET) [105]. The case which is most relevant to this work is that where an image and a standardized anatomical atlas are registered for the purpose of image segmentation: refer to section 2.3.3 on page 64 for details.

This section describes the fundamental ideas related to image registration with a focus on the particular methods employed in this work.

2.4.1.1 Image Registration Overview.

Registration is the alignment of a 'moving', 'source' or 'sensed' image S with a 'fixed' or 'target' image T. The registration process consists of four stages: *feature detection*; *feature matching*; *transform model estimation*; and *image re-sampling and transformation* [186]. Registration can be achieved by using intensity- or feature-based approaches.

With feature-based registration, objects in the image must first be identified such as edges, regions and intersections. Correspondence between these features in the images is established using similarity measures or spatial relationships. Intensity-based methods compare intensities on a pixel-wise basis between images using a sliding window approach. Correspondence between the images is found using similarity or correlation metrics. The parameters of the mapping function that aligns corresponding points between the two images are estimated and then used to perform the transformation.

The images are usually preprocessed to remove noise and ensure some consistent scale. Where pixel sizes differ between the images, one image is resampled to the scale of the other. Segmentation can also be done to aide feature extraction.

An overview of the registration process is provided here.

1. Feature Extraction. Features are extracted from the images between which some correspondence is established. Features can be simple points that describe corners, lines, or curves, or they can be more complex such as templates, regions, and patches.

2. Feature Correspondence. Correspondence is the process of matching features between the source and target images. Each set of corresponding features is assigned at least one set of points which will inform the registration transformation. The correspondence between features informs the transformation function that is applied to the source image in order to align it with the target.

3. Transformation Function Estimation. The source image S is transformed to the geometry of T by a transformation function $\mathcal{T}$. The type of transformation function used depends on the type of geometric difference between the images. It is unlikely that the geometric difference between the images is already known, so some common transformations are evaluated and iteratively updated.

4. Resampling. The transformation function is applied to S and the result is resampled onto the geometry of T. Care must be taken during resampling to avoid the introduction of artifacts into the transformed image.

Methods for image registration can be viewed as different combinations of choices for the feature space, search space, search strategy, and similarity metric [19].

The registration problem is summarized in equation (2.59). The process aims to find the spatial transformations f_x and f_y necessary to move the points (x_i, y_i) in the source image to the locations (X_i, Y_i) in the target image. This section lays out some methods for extracting these points, matching them up and transforming them.

$$X_i = f_x(x_i, y_i),\ i = 1, ..., N \tag{2.58}$$

$$Y_i = f_y(x_i, y_i),\ i = 1, ..., N \tag{2.59}$$

2.4.1.2 Feature Extraction.

In order to align two images, comparable objects or features must be found between them. Features can be area- or region-based, but they should be assigned at least one set of Control Point (CP) co-ordinates in order to inform the transformation parameters for registration.

A. Template features.

Small, predefined patches, or 'templates' represent some important detail within an image. They are typical of objects or features that are likely to be present in both images. A template may be a

real patch from the source image that is to be found in the target. The CPs for template features are set at the central pixel at matching locations in the images.

B. Region-based features.

Also called 'area features', these are high-contrast, closed-boundary regions in the images such as buildings or bodies of water. The CPs for such areas are represented by their Center of Mass (CoM) which is rotation-, scale- and skew-invariant, and is insensitive to random noise. These features can be found using segmentation methods.

C. Line features.

Lines features may be those corresponding to roads, or elongated boundaries such as that between sea and sky. Corners may also be detected. They are represented as pairs of co-ordinates indicating the beginning and end of different lines. Similarly, general edge-detection algorithms can be used to identify irregular line features that may correspond between images. CPs for line features may be the intersections of corresponding line pairs.

D. Curve features.

Curve features can fit to irregular shapes better than line features. The local maximum of curvature is set as the CP.

2.4.1.3 Feature Correspondence.

Feature correspondence is the process of determining links between features extracted from one image to locations in the other image. The methods employed depends on the features that are being used.In feature-based registration, correspondence is found between the control points identified during feature extraction. Intensity-based correspondence relies on metrics derived directly from the pixel-intensity values.

A. Feature-based Correspondence: Point Matching.

Most methods of finding correspondence rely on finding the transformation that minimizes some error or distance between CPs. The points can be found from shape, region or template features as described above, but sometimes singular points themselves are extracted. Some methods of determining correspondence between point features are described here.

A.1. *Scene Matching.*

To find correspondence between CPs in source image S and target image T, three non-colinear point pairs must be selected. A transformation $\mathcal{T}$ is obtained for the points. $\mathcal{T}$ is applied to all other points in the sets and the distances between CPs is measured. If most other points fall within a small distance of each other, then $\mathcal{T}$ is a good transformation. The three point pairs that produce the most correspondences determine the parameters of $\mathcal{T}$.

Points within a small distance of each other after $\mathcal{T}$ is applied are corresponding points.

A.2. *Clustering.*

Transformations for many combinations of three point pairs is estimated and the parameters for each set are recorded. The parameters that get the most 'votes' are the ones that define a transformation which fits the most CPs. Corresponding points are those that fall within a small distance of each other. The range of the parameters for the transformation need to be defined in this method.

A.3. *Invariance.*

There is a linear relationship between the position of three known non-colinear points and the remaining points in the image. For rigid or affine transformations (section 2.4.1.4 on page 77) this relationship is invariant after transformation. The relationship between candidate correspondence point triplets can be determined for both images and compared with a similarity metric. The difference will be small if the three points correspond with each other.

B. Feature-based Correspondence: Shape Matching.

Shape matching involves measuring the similarity between some representations of the boundary. If shapes are found with high similarity, the correspondence points are usually assigned to the Center of Gravity (CoG) of the shape.

B.1. *Fourier Transform.*

The pixels of a closed shape boundary form a cyclic sequence which represents uniformly spaced samples from a periodic signal. The Fourier Transform (FT) represents this signal in terms of its frequency characteristics. The similarity between shapes can be determined by the Correlation Coefficient (CC) between the magnitude of their FT coefficients c_k in the discrete FT shown in equation (2.60), where $z_i = x_i + jy_i$ is the complex number representing each of N points $(x_i, y_i) : i = 0, ..., N-1$ and $j = \sqrt{-1}$. The set of c_k are the *Fourier descriptors* of the shape [121].

$$c_k = \frac{1}{N} \sum_{i=0}^{N-1} z_i \exp\left(-j2\pi ki/N\right), \; k = 0, ..., N-1 \tag{2.60}$$

The magnitudes of the FT coefficients are invariant to shifts and rotations in the input z_i. By ignoring the higher-order coefficients of the FT, the effect of high-frequency noise in the image can be reduced.

Phase Correlation [96] relies on the translational properties of the FT to recover shifted images. Extensions of the method can be applied to recover rotation and scaling transforms [130].

B.2. *Moment Invariants.*

For the image with pixel intensities $I(x, y)$, some useful properties can be described by the *raw* image moments M_{ij} as calculated with equation (2.61). The *area* for a binary image or segmentation is the zeroth moment M_{00} with its centroid at $(x_0, y_0) = (M_{10}/M_{00}, M_{01}/M_{00})$, where the first raw moments M_{01} and M_{10} describe the *means*.

Raw Moment:
$$M_{ij} = \sum_x \sum_y x^i y^i I(x, y) \tag{2.61}$$

The *central* moments, or 'moments about the mean', μ_{pq} in equation (2.62) are the raw moments calculated with respect to the CoG at (x_0, y_0). The zeroth central moment $\mu_{00} = M_{00}$ represents the area, and the first central moments $\mu_{01} = \mu_{10} = 0$. The second central moment μ_{11} is equivalent to *variance*.

Central Moment:
$$\mu_{pq} = \sum_x \sum_y (x - x_0)^p (y - y_0)^q I(x, y) \tag{2.62}$$

Standardized moments normalize an n^{th} central moment by μ_{00}^n. The third and fourth standardized moments define *skewness* and *kurtosis* respectively.

Standardized Moment:
$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{1+\frac{p+q}{2}}}, \; i + j \geq 2 \tag{2.63}$$

In these discrete forms, central moments are approximately invariant to translation, whilst standardized moments are approximately invariant to scale. Rotation invariants, referred to as 'Hu moment invariants' [78], can be constructed through combinations of other moments. These translation, scale and rotation moment invariants can be used to identify correspondence between features of two images, for example, by measuring the distance between vectors of moment invariants representing the shapes.

B.3. *Shape Matrices.*

The scale of corresponding shapes between images may be different. One way to combat this is to represent the shape as a shape matrix. From its centroid, an axis is extended to the outermost pixel belonging to the shape. The shape is then resampled at regular radial and angular intervals, corresponding to the height and width of the resulting *shape matrix*. The radial intervals are a function of the maximum radius of the shape making the representation independent of the scale of the shape. Boundary information is naturally encoded within the shape matrix.

An example is shown in figure 2.12 on the next page. Similarity between two shape matrices

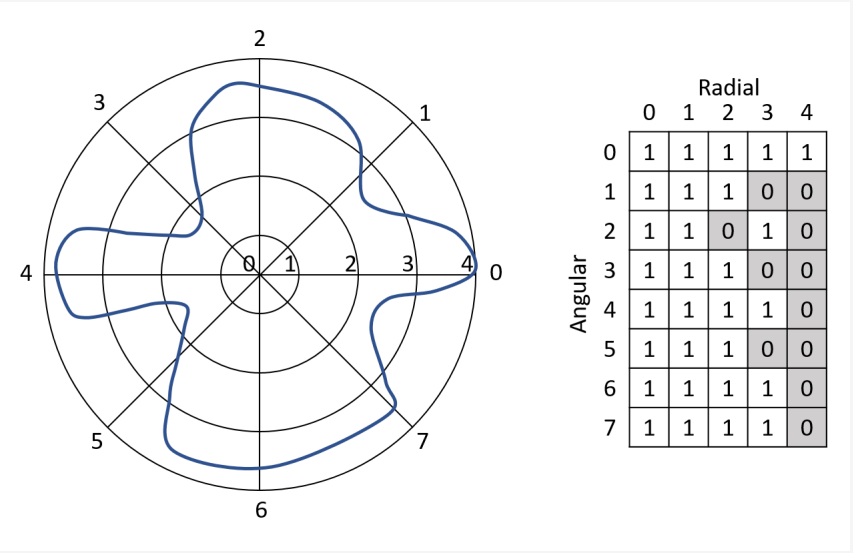


Figure 2.12: Example of Shape Matrix Construction. (left) An image on constructed polar co-ordinate axis with $\theta = 0$ aligned with the direction indicating the pixel furthest away from the shape's centroid. The shape is regularly sampled radially and by angle to create the shape matrix (right).

is found with logical operations between its binary elements.

C. Intensity-based Correspondence.

Where intensity-based features are used, image patches taken from the source image S must be matched to a patch in the target T. In 'template matching', a predefined patch may be used in order to find corrresponding points in both images. Methods such as cross-correlation and Mutual Information (MI) outlined below may be used to determine the correspondence between candidate regions in the images, as well as using the concept of moments described earlier. These methods assume that the primary transformation between two images is translation, with minimal contribution from rotation, scaling and local differences. Though using circular, rather than polygonal patches, may achieve rotation invariance. Such methods may be used to register images at a global level without the feature extraction step.

Matching intensity-based features usually requires some sliding window being passed over the images. Different measures can be calculated within the windows and compared for similarity between two images within those regions. Commonly seen metrics are described here.

C.1. *Cross-correlation.*

For a region or template $w(x, y)$ of size $K \times L$ and an image $f(x, y)$ of size $M \times N$, where $K \leq M$ and $L \leq N$, the correlation between $w(x, y)$ and $f(x, y)$ at the point (i, j) is calculated as in equation (2.65) on the following page [50] where $i = 0, ..., M - 1$, $j = 0, ..., N - 1$ and the summation is over the overlapping region. Correlation is calculated at

every window in the source S and target T images. The pair that maximizes the correlation are set as corresponding regions. Correlation can be calculated by the convolution of the region, *c.f.* filter or kernel, with the image. Details about convolution can be found in section 2.2.3 on page 40.

Correlation:
$$C(i,j) = \sum_{x=0}^{L-1}\sum_{y=0}^{K-1} w(x,y) f(x+i, y+j) \tag{2.64}$$
$$= \sum_{w} w(x,y) f(x+i, y+j) \tag{2.65}$$

Correlation is sensitive to intensity changes within the image and template, thus the normalized correlation in equation (2.66) is calculated in practice, where $\overline{w}$ is the average intensity in the region $w(x,y)$, and $\overline{f(i,j)}$ is the average image intensity in the region covered by w. This equates to the Pearson Product-Moment Correlation Coefficient (PPMCC) [50] often referred to directly as the CC.

$$NC(i,j) = \frac{\sum_w \Big(w(x,y) - \overline{w}\Big)\Big(f(x+i,y+j) - \overline{f(i,j)}\Big)}{\sqrt{\sum_w \Big(w(x,y) - \overline{w}\Big)^2}\sqrt{\sum_w \Big(f(x+i,y+j) - \overline{f(i,j)}\Big)^2}} = \frac{\sigma_{wf}}{\sigma_w \sigma_f} \tag{2.66}$$

The correlation measure can be expensive to compute on large or 3D images. Using absolute intensity differences like the ℓ_1-norm can be faster, but less accurate, when paired with a sequential search strategy outlined in [152]. The gradient of image intensities has also been shown to produce better matching results [155].

C.2. *Mutual Information.*

The similarity between a template t and a window w in an image can be determined by the Mutual Information (MI) defined in equation (2.67), where $P_t(a)$ and $P_w(b)$ are the probabilities that the intensity of a pixel in t and w are a and b respectively. $P_{tw}(a,b)$ is the joint probability that corresponding pixels in t and w have intensities a and b respectively. The sum is over all intensity values in t and w.

Mutual Information:
$$\mathrm{MI}(T,W) = \sum_{a=0}^{\max a}\sum_{b=0}^{\max b} P_{tw}(a,b) \log \frac{P_{tw}(a,b)}{P_t(a)P_w(b)} \tag{2.67}$$

A 2D histogram is constructed by plotting the histograms of t and w against each other then binning the plot to achieve a 2D array. The entries of this array correspond to $P_{tw}(a,b)$. $P_t(a)$ and $P_w(b)$ can be estimated by the sums over row a and column b respectively. An example is shown in figure 2.13 on the following page

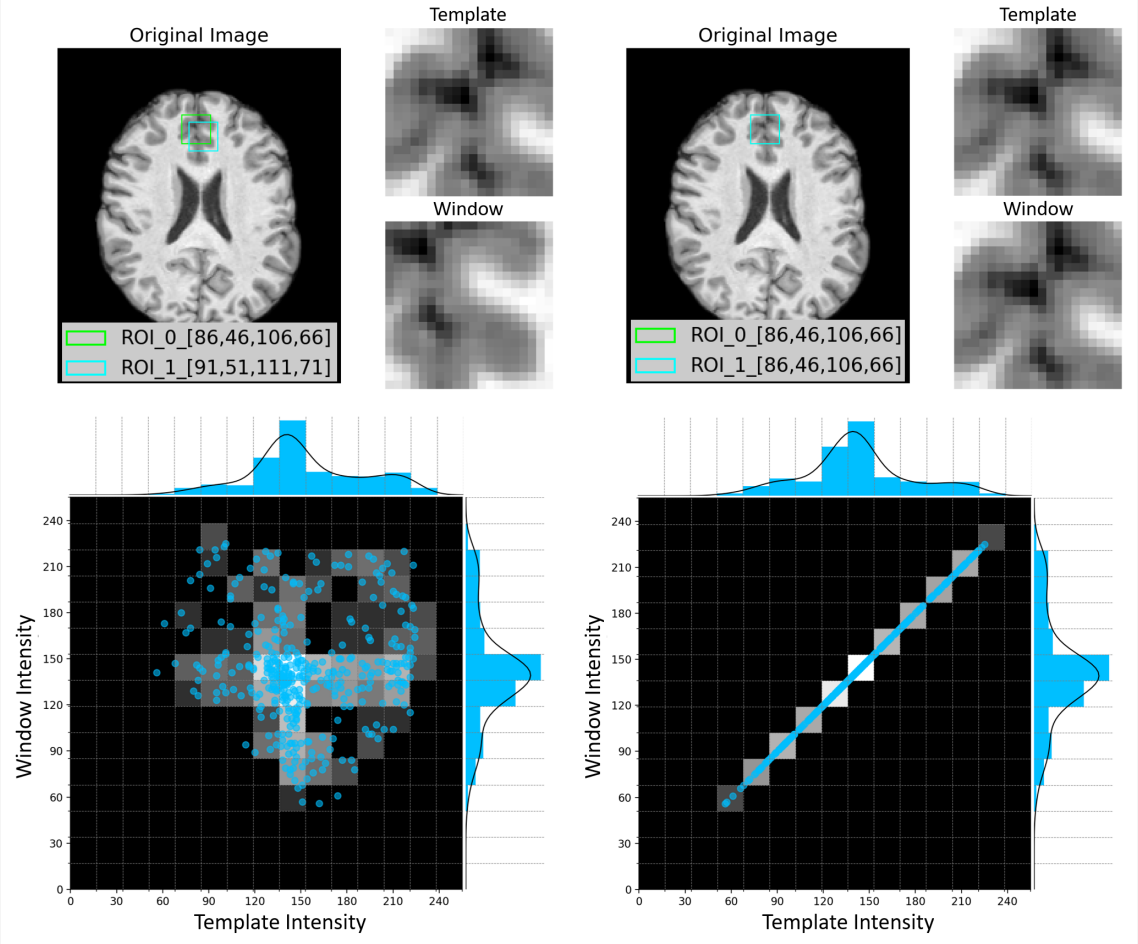


Figure 2.13: Template and Window Joint Intensity Distribution (Top-left) The original image showing the location of an extracted template (cyan) and a slightly shifted window under test (green). The template and window are shown. (Top-right) The template and window where both coincide. (Bottom). The joint- and marginal- probability distributions of image intensities found through binning the continuous intensity values. The logarithm of the 2D histogram values are shown for better visualization. Images generated with the script in listing 6.4 on page 216.

Feature correspondence can be found by finding the windows in the source S and target T images that maximize the MI between the themselves and the template. The method works well when there is a consistent mapping between intensities in t and the image, but is sensitive to noise.

MI has been employed successfully for registering multimodal images, *e.g.* Magnetic Resonance (MR), CT and PET [105], and breast MRI [139].

C.3. *Kolmogorov-Smirnov Test.*

The intensity, or cumulative intensity [41], histograms for the template and the window-under-test can be examined and statistically compared to determine their correspondence. The

Kolmogorov-Smirnov (K-S) test T in equation (2.68) measures the greatest vertical distance between the template distribution $F_t(I)$ and image window distribution $F_i(I)$. The hypothesis that the samples come from the same image is rejected at some significance α when T exceeds $c(\alpha) = \sqrt{0.5 \cdot \ln(\alpha/2)}$, otherwise correspondence is set.

K-S Test: $$T = \sup_I |F_t(I) - F_i(I)| \tag{2.68}$$

C.4. *Coarse-to-fine Feature Matching.*

It may be beneficial to first register down-sampled versions of the source S and target T images [84]. Any non-linear geometric transformations that are necessary in the higher-resolution diminish at the lower scale. First, the global geometric difference between the images are determined at coarse resolution and S is resampled to achieve global alignment. The scale is gradually increased and more CPs are selected that identify correspondence between local geometric differences at finer resolution. Images should first be smoothed with a Gaussian filter.

If S and T are down-sampled by a factor of 2 to get S_2 and T_2 at half the resolution, and S_2 and T_2 can be globally registered with the functions $X = T_x(x, y)$ and $Y = T_y(x, y)$, then the same functions can be used to register S and T such that $X = 2T_x(x/2, y/2)$ and $Y = 2T_y(x/2, y/2)$.

In a similar vein, a subset of a template can be used to perform a faster identification of candidate correspondence points between the template and the image. The full template can then be employed later to determine the best of those candidates [60].

When applied to the search itself, coarse-to-fine feature matching may mean using a larger stride for the sliding window on the initial pass, then choosing locations with a higher similarity statistic as candidates for the more fine-grained search at each pixel location.

2.4.1.4 Transformation Estimation.

Where the geometric differences between two images are global, the most common image transformations to use are the rigid and affine transformations detailed here. Extensions and generalizations of these transforms are also used, such as projective and global polynomial transformations. A transformation function that accurately maps the CPs, found through feature-extraction and correspondence, in the source image S to the target image T should also map the other points in the images without causing too much distortion of S.

A. Rigid Transformations.

A rigid transformation is composed of translation T, rotation R, and scaling S operations which account for movement of the image-subject or sensor. Rigid transforms do not alter the relative

shape or size of objects within an image, straight lines in the original image are straight lines in the transformed image. They are represented as matrices which, when multiplied with the image, produce the required transformation.

A rigid transformation can be found by the similarity transform in equation (2.70) and equation (2.71), where S, R and T are the scaling, rotation and translation transformations with parameters (s_x, s_y), θ, and (t_x, t_y) respectively.

$$I' = SRI + T \tag{2.69}$$

$$X = s_x x \cos\theta - s_y y \sin\theta + t_x \tag{2.70}$$

$$Y = s_x x \sin\theta + s_y y \cos\theta + t_y \tag{2.71}$$

A.1. Translation. Translation is a uniform shift of all pixels in the image such that the pixel at $[x, y]^T$ moves by tx and ty pixels in each of the x and y directions respectively. Translation is an additive, rather than multiplicative, transformation thus *homogeneous* coordinates are used to perform the linear transformation $I' = TI$ and is shown in equation (2.72). In this system, $[x, y, w]^T$ represents a point at $[x/w, y/w]^T$ with the convention $w = 1$.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \tag{2.72}$$

A.2. Rotation. A rotation transformation $I' = RI$ in Euclidean space can be performed by the multiplication of the rotation matrix with the image, shown in equation (2.73). The rotation direction is counter-clockwise for $\theta > 0$ and clockwise otherwise.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \\ 1 \end{bmatrix} \tag{2.73}$$

A.3. Scaling. Scaling is a linear transformation $I' = SI$. It enlarges all distances in a given direction by a constant factor s shown in equation (2.74). It does not affect distances in the perpendicular direction. For $s > 1$, distances are enlarged and for $s < 1$ compression or downsampling is performed. The identity transform is applied at $s = 1$.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} \tag{2.74}$$

B. Affine Transformations.

Whilst rigid transformations preserve the shape and scale of objects within an image, it may be necessary to perform transformations that do not respect these conditions. An affine transformation may not preserve the angles between lines or the distance between points, but ratios of distances between points lying on a straight line are preserved. This means transformations such as *shearing* can be applied. An affine transformation is the more general case of the similarity transform and can be written as in equation (2.75) and equation (2.76), where a-f are the elements of the matrix denoting the affine transformation parameters.

$$X = ax + by + c \tag{2.75}$$

$$Y = dx + ey + f \tag{2.76}$$

The parameters depend on the order of the transformations being applied. Transformations are associative, *e.g.* $TR(S) = T(RS)$, but non-commutative, *e.g.* $TR \neq RT$. This is exemplified in equation (2.79) with a translation of $(2, 2)$ and rotation by $\theta = 90°$.

$$R_{90}T_2 = \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.77}$$

$$T_2R_{90} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 2 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.78}$$

$$R_{90}T_2 \neq T_2R_{90} \tag{2.79}$$

B.1 Shearing. The shear transformation $I' = HI$ produces a 'slanting' of the image. It can also be defined as a matrix transformation on the image given in equation (2.80).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + h_xy \\ y + h_yx \\ 1 \end{bmatrix} \tag{2.80}$$

An affine transformation can be composed of translation, rotation, scaling and shearing operations, which can be combined into a single transformation matrix. Equation (2.84) on the next page shows the scaling $s = 2$, rotation $\theta = 90°$ and translation $t_x = 5, t_y = 1$ of a point. Rotation operations are relative to the Origin, and so the image is first translated such that the centre of rotation is at the Origin and then the inverse performed to return the image to its original position after the transformation.

$$I' = T_{(5,1)} R_{(90)} S_{(2)} I \tag{2.81}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.82}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \cos\theta & -s_y \sin\theta & t_x \\ s_x \sin\theta & s_y \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.83}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 & 5 \\ 2 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -2y + 5 \\ 2x + 1 \\ 1 \end{bmatrix} \tag{2.84}$$

Examples of affine transformations applied to a 2D slice of a 3D brain MRI are shown in figure 2.14 on the following page.

C. Extension to 3D Affine Transformation.

Transformations in 3D are simple extensions of the 2D cases. Translations are still additive and interpolation, discussed in section 2.4.1.5 on page 82 is now done in 3D. Shearing in one dimension causes a shear in the other dimensions indicated in the transformation matrix in equation (2.87). Shear factors are typically specified in the form h_a^b, where a is the axis on which shear is being applied and b is the co-ordinate axis that the factor is being applied to.

3D Translation:
$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.85}$$

3D Scaling:
$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.86}$$

3D Shearing:
$$H = \begin{bmatrix} 1 & h_x^y & h_x^z & 0 \\ h_y^x & 1 & h_y^z & 0 \\ h_z^x & h_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.87}$$

A 3D rotation is composed of a rotation around the axis in each of the 3 dimensions meaning that each of the rotation matrices in equation (2.89) on the next page can be applied sequentially.

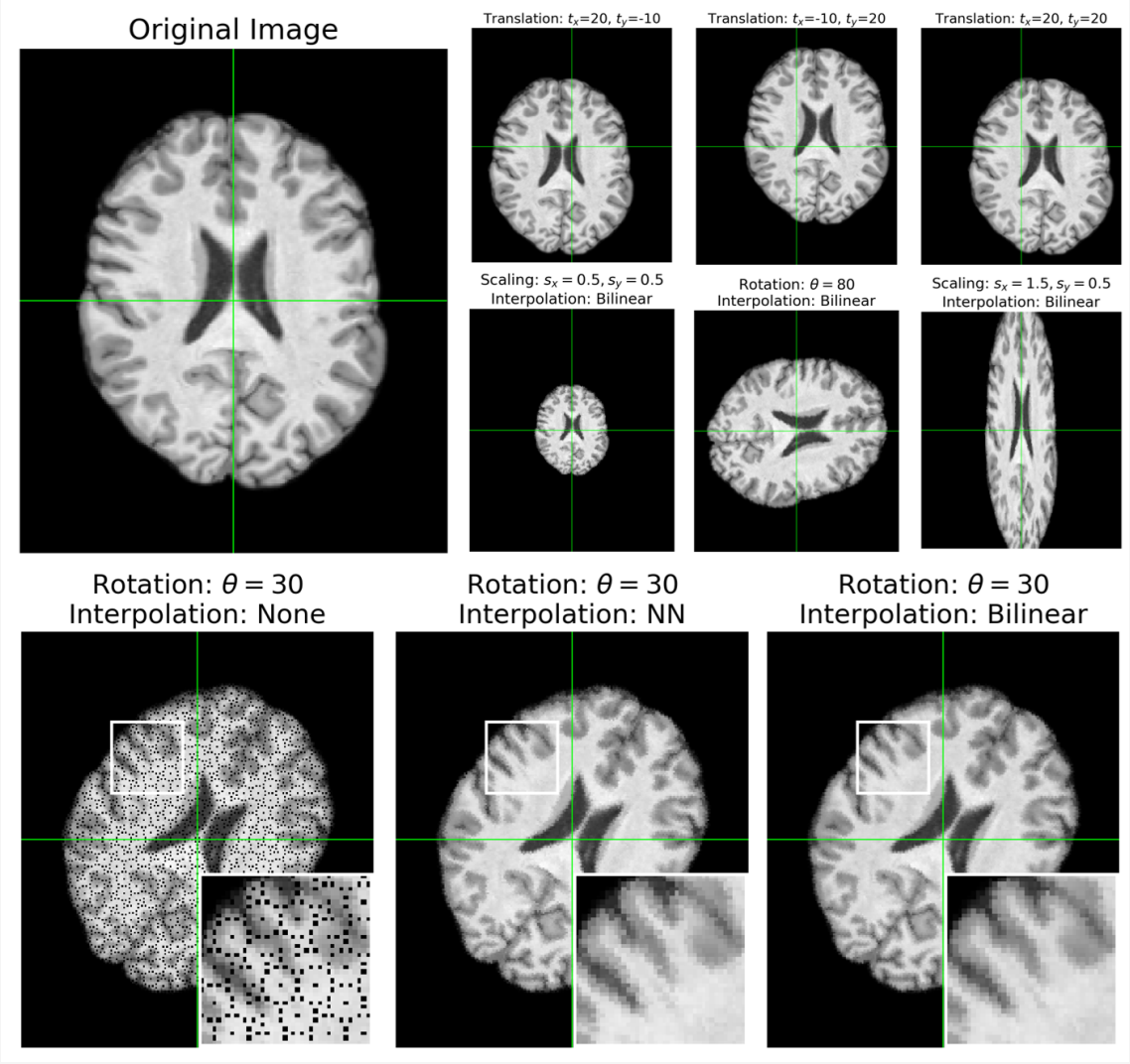


Figure 2.14: Examples of Affine Transformations. Each image is a transformation of the (top-right) original image using the parameters indicated, where s_x, s_y, t_x, t_y and θ are the scaling factors in the x and y directions, translation parameters in the x and y directions and the rotation angle respectively. Bottom panel demonstrates the effect of (left-to-right) no interpolation, Nearst-neighbour (NN) interpolation and bilinear interpolation on the image transformed by a rotation matrix. Insets show enlargement of the same area in each image. Images were produced with the script in listing 6.5 on page 221.

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma) \tag{2.88}$$

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.89}$$

To perform registration of points in the source image S to the target image T, s_x, s_y, θ, t_x and t_y need to be determined. θ is determined by the angle between the lines connecting CPs in the images and s_x and s_y is found from the ratio of distances between CPs in the images. When scaling and rotation parameters are found, t_x and t_y can be determined. At least three non-colinear corresponding CPs are needed. A least-squares fit is applied such that sum of squared errors at the CPs is minimized.

D. Non-linear Transformations.

Images with small, negligible non-linear geometric differences can be well-registered by linear transformations, otherwise a composite of local geometric transformations may be needed to successfully align the images. The least squares approach to fitting transformation parameters averages out local geometric differences over the entire image, which may not be desirable for image-subjects prone to non-liner distortions, as is usually the case in medical imaging.

A weighted least squares or weighted mean method [62] may be applied to limit the influence of individual CPs over a particular region. Piecewise linear [61] and cubic mappings apply local mapping functions which are valid only for individual patches in the image. Radial Basis Functions (RBFs), whose values depend on the distance from their centers, can also be employed. Gaussians, multi-quadratics and thin-plate splines are representative of such RBFs.

2.4.1.5 Resampling.

The transformation function $f(x, y)$ takes integer-valued Control Point (CP) locations (x, y) in the source image $S(x, y)$ and transforms them to floating-point co-ordinates (X, Y) in the target geometry $T(x, y)$ creating the transformed image $S'(X, Y)$. The pixel intensity in the target geometry must be found by resampling S' to the regular grid.

The resampling can be done naïvely by taking the nearest discrete pixel location to each continuously valued co-ordinate and assigning the pixel intensity from S'. However, this can lead to errors where no points in S' lie within a pixel in the geometry of T. Figure 2.14 on the preceding page shows clearly that rotating an image grid can lead to 'holes' in the transformed image when the discrete pixel intensities were inferred from the transformed co-ordinates. The same occurs when images are scaled. Such 'empty' pixels can be avoided by resampling using intensity values from nearby pixels, known as 'interpolation'.

A. Nearest Neighbour Interpolation. Nearst-neighbour (NN) is a simple interpolation method whereby the intensity value of a pixel is determined by the value of the closest point in the transformed grid. The resulting image may appear jagged, particularly at edges, see figure 2.14 on the previous page. With the NN method, new intensity values are not created so histograms of the image before and resampling remain similar.

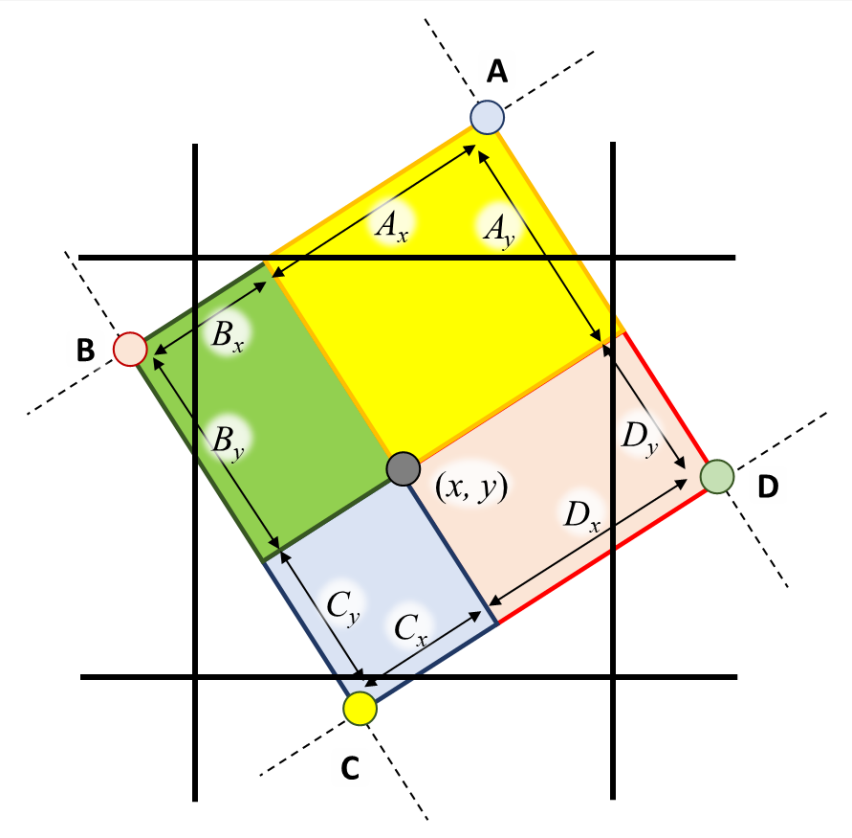


Figure 2.15: Bilinear Interpolation. No point in the transformed (dashed) grid lies within the pixel $(x,\ y)$. The intensity value at (x, y) is the weighted sum of the intensities at the points A, B, C and D on the transformed grid. The weight of each intensity corresponds to the opposing area to the point on the transformed grid, calculated by equation (2.90).

B. Bilinear Interpolation. A more complex method of interpolation is the bilinear approach. Here, each of the points on the transformed grid surrounding the empty pixel contributes to its intensity value. Points closer to the pixel location are weighted more. For a transformed grid whose neighbouring points A, B, C and D do not lie within the pixel $(x,\ y)$, the bilinear interpolation result can be found with equation equation (2.90), where $Z_i, Z \in \{A, B, C, D\}, i \in \{x, y\}$ are the x and y components of the pixel centroid's displacement from each neighbour.

$$I(x, y) = A_x A_y I(C) + B_x B_y I(D) + C_x C_y I(A) + D_x D_y I(B) \tag{2.90}$$

Bilinear interpolation has a smoothing or blurring effect on the image and creates new intensity values, thus the histograms of the images before and after may appear dissimilar.

2.5 Challenges to the Deployment of Machine Learning Models for Medical Image Analysis.

2.5.1 Introduction

Medical imaging is the field in which visual representations of the internal structure and function of the body are acquired in order to aid clinical analysis. Medical imaging data are often large, heterogeneous, multi-dimensional arrays which can be dense or sparsely populated.

Such data have posed substantial challenges to the training of Machine Learning (ML) algorithms due to their size and heterogeneity. However, the distinct advantages of objective and reproducible ML-assisted diagnosis and treatment planning has spurred an explosion in the development of 'Artificial Intelligence (AI)-enabled' medical image analysis.

There are substantial challenges to the development and clinical deployment of ML-algorithms.

Challenge 1: Training Data Quality.

Datasets used to train ML algorithms should be sufficiently large, well-curated, and, crucially, representative of the population of images on which they are to be evaluated. Obtaining annotations or Ground Truth (GT) labels for these data is particularly challenging due to the time, effort, knowledge and cost involved.

Challenge 2: Prediction Quality Control (QC).

It is important that clinicians, patients and other stakeholders have confidence in the predictions made by ML models that have been deployed. Providing assurance of the prediction quality is imperative to improving healthcare and building trust between researchers, practitioners and the general public.

Challenge 3: Algorithmic Bias.

Datasets have the potential to contain intrinsic biases, whether that be based on demographic differences such as the sex or age of the subjects they contain, or due to the mode of image acquisition, *e.g.* Magnetic Resonance Imaging (MRI) or Computed Tomography (CT). ML algorithms can adopt these biases causing a degradation in performance between datasets.

Though it is not the focus of this work, Challenge 1 is important to consider here. The availability and calibre of training data is dependent on many factors. Each scanner is different; each operator is different; and each scan is conducted with a particular clinical focus and on the individual subject's needs. This leads to highly heterogeneous data ranging from high quality, ideal

images to those containing artifacts, missing slices, and corrupt header information. The format in which the data are saved also varies. Inconsistent terminology and intra- and inter-institutional convention differences may also exist. Beyond the images themselves, the laborious and time-consuming nature of image annotation is in conflict with the availability of experts who can perform the task.

Where medical imaging data are acquired and curated for use in ML models, they often form small datasets which may not be wholly representative of the data on which the models will be evaluated. Large-scale population imaging studies such as the UK Biobank (UKBB) [154] aim to standardize a plethora of medical imaging procedures whilst scanning tens of thousands of subjects. Population studies like this are proving to be highly fruitful due to the sheer volume of data, but still suffer from the lack of expert annotations. Good quality automated ML models are used to provide labels in this situation, but assuring the quality of these is a challenge in itself. Preparing medical imaging data for ML is an area of ongoing research [173].

This work considers Challenges 2 and 3. It focuses on the QC of predictions made by ML-based medical image analysis tools. Specifically, it centers on the quality of automatically generated biomedical image segmentations with a focus on cardiac MRI. This work also studies the problem of bias in ML algorithms focusing on the Domain Shift (DS) problem where ML algorithms trained to perform well on one dataset may not necessarily perform as well when the data comes from a different site or scanner. This effect is explored using neuroimaging data from multiple sites.

2.5.2 QC of Machine Learning Predictions on Medical Images.

Quality Control (QC) may mean different things in different contexts, with the term often used interchangeably with Quality Assurance (QA) in literature. For clarity, QA deals primarily with the *process* that generates some outcome, whereas QC is focused on the *outcome* itself. In this work, QC is used to mean the process of assessing some outcome in order to ensure that it is fit for purpose. This work focuses on QC of cardiac Magnetic Resonance (MR) segmentations which asks whether segmentation S of image I is accurate *enough* in order to be used in subsequent analyses, *e.g.* automated Left Ventricular Cavity (LVC) volume calculations.

The process of QC is traditionally done manually by human observers or 'raters' who check human- or machine-created predictions. Problems can arise in this regime such as inconsistency between assessments due to *intra*-rater variation, which may, understandably, be caused by fatigue. This can be overcome by performing multi-rater assessment of a prediction, but this can lead to *inter*-rater variation, perhaps caused by disparity between rater experience. The manual QC process becomes infeasible when the number of predictions to be assessed becomes very large.

In [88], three creator-verifier schemes are laid out: human-human, machine-human, and human-machine. The first two schemes suffer from the same QC challenges outlined above; namely sub-

jectivity, human-error and lack of scalability. This has led to a surge in research on methods that aim to improve the machine-verification task such that the QC process for predictions becomes automated.

It should be noted that during the training process, ML models undergo training, validation and testing stages where the training data is split into correspondingly named partitions of randomly selected subsets and only the training set is used during learning. The validation set is used to indicate the generalization of the model to unseen data during training. The training and validation sets may be joined and re-partitioned in order to perform cross-validation on the model. The test set is kept separate and is only used to assess the final performance of the model. This process may be considered as a form of internal QC for ML models, but it is different to the QC being discussed in this work.

A well-trained ML model will yield good quality results when evaluated on a test set, but there may be specific instances in the test set for which the model prediction fails. The reason for this failure could be many-fold: the datapoint is an outlier and is unrepresentative of the rest of training set; the datapoint is drawn from a different distribution; or the datapoint is corrupt *i.e.* it contains artifacts or high levels of noise.

In the context of this work, QC refers to assessing predictions of a ML model in order to identify when it is failing (chapter 3 on page 88). Moreover, this work focuses on measuring the quality of ML predictions on a *per-case* or *per-instance* basis, rather than assessing the average quality of the model predictions.

A final creator-verifier scheme, machine-machine, may be considered, whereby predictions are generated by a ML model and their quality also assessed by a ML model. This is the scenario in automated image analysis pipelines such as those in clinical trials. It may also be envisaged that such a setup would be useful for the analysis of large-scale imaging databases, like the UKBB. Whilst some methods of automated QC assess batches of predictions at regular time intervals, this approach does not provide an informative account of prediction quality on a per-case basis.

The result of a QC procedure may inform other processes. Over time, if the ML predictions start to drift, perhaps due to changes in the population data under test, an automated QC assessment may feedback to the predictor model where extra checks and updates can be made. It may also be possible to provide instantaneous, real-time feedback to users of a deployed model such that they can make informed and confident choices regarding their next steps. For example, a real-time QC procedure on ultrasound image analysis, or an MRI segmentation algorithm, may instruct the operator to reacquire the images to gain increased quality in the resulting analysis.

2.5.3 Bias in Machine Learning with Medical Imaging Data

When training a ML model, the training dataset should be representative of the data on which it is to be evaluated [57]. The process of collecting samples and assigning labels is laborious, but possible assuming that the test distribution is known before training is done. However, avoiding the unconscious introduction of bias into the training set is non-trivial, particularly in medical imaging where there exist a number of variables which may easily be overlooked.

Common sources of bias in medical imaging datasets may include gender, age, and ethnicity [2] imbalances as well as the prevalence of some pathology. Datasets that are collected by a single site, perhaps as part of some study, may be biased by the socioeconomic [54] and environmental factors that are particular to the surrounding region. These factors, along with the presence of confounders, can lead to an under- or over-performance of ML models such as predicting hospital 'length of stay' [44]. Model performance differences that occur due to demographic changes cause a collective *population shift* discussed in section 5.2.2 on page 140.

Combating dataset biases requires collecting large volumes of data from across a range of providers in a standardized way. Some large-scale studies exist such as the UKBB and the Observational Health Data Sciences and Informatics (OHDSI) initiative [77].

Assuming that demographic biases are minimized, and that the images show the same anatomy, intrinsic biases caused by imaging scanners also threaten model performance. Differences in MRI field strength and image reconstruction processes may cause variations in the raw and post-processed data that are not easily discernible to the human eye. Studies of multi-center imaging data have shown that gradient distortion corrections need to be applied in order to reproduce MRI scans of the same subject in different scanners [86]. The decision whether to acquire a 2D-stack or full 3D image can affect image quantification [66] due to slice thicknesses. Model performance differences that occur due to scanner effects cause a collective *acquisition shift* which is discussed further in section 5.2.2 on page 140.

In medical image analysis, care must be taken to ensure that every possible bias is removed from the training set such that the model is better able to generalize to unseen data. However, certain intrinsic bias such as scanner effects, are less easy to decouple from the image features which may be important for the task for which a model is being developed. The work in chapter 5 on page 136 focuses on methods to counteract such bias.

Chapter 3

Automatic Quality Control of Cardiac Magnetic Resonance Image Segmentation in Large-scale Population Imaging

A Validation of Reverse Classification Accuracy

The trend towards large-scale studies, including population imaging, poses new challenges in terms of Quality Control (QC). This is a particular issue when automatic processing tools such as image segmentation methods are employed to derive quantitative measures or biomarkers for further analyses. Manual inspection and visual QC of each segmentation result is not feasible at large scale. However, it is important to be able to detect when an automatic method fails in order to avoid inclusion of wrong measurements into subsequent analyses which could otherwise lead to incorrect conclusions. To overcome this challenge, this work explores an approach for predicting segmentation quality based on Reverse Classification Accuracy (RCA), which facilitates discrimination between successful and failed cases. The approach is validated on a large cohort of Cardiac Magnetic Resonance Imaging (CMR) for which manual QC scores are available. Results on 7,425 cases demonstrate the potential for fully automatic QC in the context of large-scale population imaging as in the UK Biobank (UKBB) Imaging Study.

The work in this chapter was published in the proceedings of the 20th annual conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), September 2017 [133] and the Journal of Cardiovascular Magnetic Resonance (JCMR) [134].

3.1 Overview.

Biomedical imaging data are increasingly processed with automated image analysis pipelines which employ a variety of tools to extract clinically useful information. There are often many modules in automated pipelines [150] where each may contribute to inaccuracies in the final output and reduce the overall quality of the analysis. Examples of such modules may include: intensity normalization to some predefined range; spatial normalization by registration; segmentation to gather metrics on the scene; and feature extraction for further analysis. It is important to understand the limitations of such pipelines and assess the quality of the results being reported. However, each part of the pipeline poses its own challenges to quality assessment.

This is a particular issue in the context of large-scale population imaging databases comprising thousands of images, such as that of the UK Biobank (UKBB) Imaging Study [154]. On a large scale, it is infeasible to perform a manual, visual inspection of all outputs, and even more difficult to perform Quality Control (QC) within the pipeline itself. The issue also arises in online image processing systems where new data may be constantly streamed into the pipeline from external sources. The imaging data uploaded from clinical trials may be a pertinent example.

This chapter concerns automated QC in these large-scale environments containing thousands of images. The focus here is on the specific challenge of automated QC of biomedical image segmentations. In particular, the problem of automated QC on a per-case basis in the absence of Ground Truth (GT) reference segmentations is considered.

Section 3.2 on the next page introduces the problem of automatic evaluation of biomedical image segmentations. An overview of the research conducted in this domain is presented in section 3.3 on page 92. The methods and datasets employed in this work are outlined in section 3.4 on page 96 with results shown in section 3.5 on page 106. Section 3.6 on page 115 gives a discussion on the implications of these results for future work on automated QC.

3.2 Introduction.

A segmentation may be required for a number of reasons: counting the number of objects belonging to the same class; obtaining some geometric measurements of particular objects or regions (*e.g.*, area or perimeter); or studying some statistical properties of an individual object such as its texture. However, segmentation is an ill-posed problem because there are many solutions depending on the segmentation target and method of segmentation used.

Many image segmentation methods have been developed, from simple edge-detection and image tresholding techniques to more advanced algorithms involving machine learning. These are outlined in section 2.3.4. Each method may generate a different result. Indeed, applying the same segmentation method many times may also yield different results depending on initial conditions, such as seed points. Therefore it is difficult to know which segmentation result is the 'correct' one.

Traditionally, segmentation methods are evaluated on a set of labelled data using an evaluation metric which computes similarity between the predicted segmentation and a reference GT. Popular metrics include volumetric overlap, *e.g.* Dice Similarity Coefficient (DSC), surface distances or other statistical measures discussed in section 3.4.2 on page 99.

In the medical imaging domain, there is often a lack of actual GT reference segmentations. Manual expert annotations are often used as the reference. However, many studies have shown that even for expert annotators, there is some amount of *inter*-rater variability between different experts. That is, different experts will produce different segmentations. Furthermore, *intra*-rater variation exists when an expert is asked to segment the same image multiple times.

Models for medical image analysis are usually developed offline in some research environment. They are trained and tested on datasets which are unlikely to come from the specific environment in which they are to be deployed. Thus, it is likely that there will be some variability in the performance of the algorithm due to some form of Domain Shift (DS); chapter 5 explores the problem of DS in more detail.

The algorithms are ultimately deployed in some software or automated pipeline. It is important to regularly assess the quality of the segmentations produced by the model in order to ensure it is performing effectively. This may be done periodically by evaluating the model on some images for which manual expert segmentations have been obtained. However, once a segmentation method is deployed in clinical practice, GT labelmaps or expert annotations are demanding and costly to acquire. This makes it difficult to evaluate the performance of the algorithm over time.

Evaluating the expected average performance of an algorithm is less important than being able to assess the quality on a per-case basis. It is crucial to identify cases where a segmentation has failed to avoid mistakes in downstream analysis tasks.

3.2.1 Contributions.

This work explores a reverse testing method, based on Reverse Classification Accuracy (RCA) [164], for the effective per-case prediction of segmentation quality in 3D Cardiac Magnetic Resonance Imaging (CMR). It provides a thorough validation of the algorithm for the application to fully automated QC in large-scale population studies.

First, a modified RCA is validated against a large dataset for which manual QC scores have been obtained. Results indicate highly accurate predictions of segmentation quality across various metrics on some 7,425 cases. The method is then applied to a separate CMR dataset which includes manual expert labelmaps yielding promising results for automated QC in this area. Finally, the automated QC pipeline is used to assess the quality of segmentations generated by a state-of-the-art Deep Learning (DL)-based method with good outcomes.

3.3 Related Works.

Machine Learning (ML) research is fast-paced and the state-of-the-art is constantly changing. The work in this chapter deals with methods that are unlikely to compete against DL-based approaches in terms of run-time and, to some extent, accuracy. However, these methods that do not rely on DL can often offer a much more intuitive and explainable approach which is desirable in many fields, particularly in the medical imaging domain.

The literature surveyed for the work in this chapter is limited to those that do not rely on DL methods. Instead, it focuses on those that use tried-and-tested techniques, such as registration, and non-DL ML methods such as Support Vector Machines (SVMs). Literature that deals with DL-based approaches to segmentation QC are considered in the literature survey in chapter 4, where they are more pertinent and comparable to the work presented on real-time QC in algorithm deployment. This chapter validates and evaluates a methodology which may not necessarily beat the state-of-the-art in run-time, but, without DL, is able to offer excellent accuracy and low error rates in QC at large-scale in the absence of GT.

The accuracy of a segmentation is ultimately limited by the quality of the image from which it is derived. Thus, many attempts to understand segmentation quality have focused initially on Image Quality Assessment (IQA). The quality of a segmentation generated by an automated algorithm, *e.g.* Random Forests (RFs), may be dependent on the quality of the input image. If the contrast or resolution of the image varies from what has been seen before, automated algorithms may fail. This may also be the case where images do not contain the whole anatomy on which models have been trained, as with missing slices in Magnetic Resonance Imaging (MRI) scans.

3.3.1 Image Quality Assessment.

Extensive investigations into IQA for MRI have been performed, often focusing on clinical trials [117]. Assuring the quality of the images acquired in CMR is fundamental to the accuracy of downstream analyses of cardiac function. In [21], IQA was conducted on the pilot dataset of 5,000 UKBB cardiac Cine MRI, focusing on the available cardiac views. The analysis focused on free-text and numerical quality assessment of the imaging data by multiple experts. Contours of the heart were also manually delineated thus relating the segmentation to the image quality. The manual analysis is prone to human error and does not provide a quantitative evaluation using traditional metrics. It may also be possible to have a good segmentation of an image which is of poor quality when the contours are processed manually - thus the numerical score does not directly relate to the quality of the segmentation itself.

IQA with Convolutional Neural Networks (CNNs) is discussed briefly in section 4.3 on page 121, however this work focuses on the assessment of medical image *segmentations*, rather than directly on image quality. An overview of IQA in medical image analysis is available in [27] for further

discussion on this issue.

3.3.2 Manual Segmentation Quality Control.

Assuming that the quality of the CMR images is good, automated segmentation algorithms may still generate poor quality labelmaps. A direct assessment of segmentation quality is needed.

In the case of manually annotated labelmaps, simple overlap measures have been used to assess the inter-rater variation between repeated segmentations of the same anatomy [187]. Though this alleviates the need for a GT segmentation of the images, manual segmentation of images is a laborious task for which many automated methods have been developed. It is right to take advantage of these automated methods to reduce the burden on experts, but there is still the need for some automated quality assessment of labelmaps which is tractable at large-scale and in the absence of GT.

Most manual approaches to segmentation QC rely on assigning the labelmap to some categorical scale - 'good' to 'poor'. The scale may be numerical, indicating the relative quality of segmentations across a dataset. In this work, the proposed approach is compared with results from a manual QC inspection on the same set of UKBB data. The manual scheme scored basal, mid and apical slices of CMR segmentations from 0 ('bad') to 2 ('good') giving a total score out of 6. This is a laborious task over almost 5,000 scans - 15,000 slice scores.

The QC procedure must be automated to speed up the process and to prevent errors due to, for example, rater fatigue.

3.3.3 Automated Segmentation Quality Control.

A distinction must be made between the quality of a segmentation model, and the quality of an individual segmentation produced by the model. Though a segmentor may generate excellent labelmaps on one dataset, it is not guaranteed to perform as well on others. When deployed, the quality of generated segmentations may deteriorate due to changes in the population of subjects being imaged as well as changes in imaging devices: chapter 5 investigates this more closely.

Reverse validation [182] is a method based on the idea of reverse testing [47]. To assess the quality of a model, *e.g.* a classifier, a new model is trained on the predictions on the test data which is then evaluated on the original training data. If the new model is unable to give predictions which correspond to the training data GT, then the original model may not be producing correct predictions on the test set. As the models are trained on whole sets of images, reverse validation is only able to give an estimate of average model performance.

The quality of a set of segmentations can be assessed to estimate the overall average performance of a segmentor. This has been studied in the context of deployed medical image analysis pipelines.

In this chapter, this work focuses on the quality of *individual* segmentations. The aim is to provide some measure of segmentation quality on a case-by-case basis.

In some applications, simple approaches such as template matching [45] may be an appropriate way to measure segmentation quality alongside traditional metrics such as overlap. However, in medical image analysis, though there may be a certain level of structural consistency between subjects, there is a large amount of variation in the images acquired. Segmentations of anatomy will not fit to pre-defined templates making this method non viable for segmentation QC in medical image analysis. Some of the other methods described here may use reference sets of images which may act as 'templates' under some warping criteria.

The Adaptive Composition of Reference Segmentations described by [120], segments an input image by stitching together pieces of labelmaps from a reference set of images with multiple manual annotations per image. The manual labelmaps differ holistically, but tend to be locally consistent, thus a composition created from these segmentations will capture a greater level of human perception than a single annotation. The underlying assumption is that if an automated segmentation is good, it can be composed by pieces of the manual annotations in the reference set. The work focuses on the natural image domain, though it can be compared with multi-atlas segmentation methods which have been applied in medical image analysis. As such, the idea is not unlike the work in [164] which also uses a reference set of manual annotations, but lacks the compositional approach.

In [25], two measures of segmentation quality are proposed. First an 'objective' discrepancy parameter considers the spatial accuracy of the labelmap based on False Positive (FP) and False Negative (FN) pixels with respect to a reference. The second is a 'perceptual' error which considers spatial and temporal changes to a labelmap. It applies a weighting to erroneously labelled background pixels and includes 'surprise' and 'fatigue' metrics in the temporal case. The overall quality prediction is compared with human raters. The metric is useful for applications where the final judge of the quality is a human observer or the results of segmentation are otherwise processed in a human-like fashion. However, in most modern applications, automated labelmaps are used to derive important measures from images which may be used in subsequent analyses, *e.g.* cardiac structure and function. Therefore is it not appropriate for use at large scale.

Optimization-based automated segmentation algorithms minimize some objective function comprised of numerous components. In [93], 42 features are taken from numerous energy-based algorithms, including geometric, intensity, and gradient features. These features were used to train a SVM-based (see section 2.2.2.2 on page 35) regressor to predict several segmentation error metrics with respect to GT. In this approach, as with most ML works, a critical limitation is the availability of a large number of training samples for which GT is available. Additionally, the classification of 'good' and 'poor' segmentations in [93] are done at a volumetric overlap threshold of only 10%. In cases where algorithms are deployed, it is desirable to identify segmentations of poor

quality as well as those which have completely failed. This requires a continuum of segmentation quality scores based on examples.

These previous approaches are not directly applicable to predicting segmentation quality at large scale on a per-case basis in the absence of GT. This work adopts the approach of RCA first proposed in 2017 in [164].

Work in [71] is a precursor to RCA. It proposes a method of atlas selection for use in multi-atlas segmentation models. The primary contribution of the work is the proposal of a 'mirror' or backward registration of the segmentation target to the atlas space. A measure of overlap is calculated to identify correlation between target segmentation in the forward direction and atlas GT overlap in the backward direction. It is a judge of 'registration success'. The method is used to select altases which may be better at performing segmentation of different structures and classes in the target domain.

RCA [164] builds on the preliminary work in [71]. It is a general framework for predicting the performance of a segmentation method on a case-by-case basis when the GT is unknown. In contrast to previous approaches for predicting segmentation quality, RCA has the advantage of not requiring a large training database. Nor does it require varying degrees of segmentation quality for the GT. A reference set of segmentations is used in this process. Using a reference set, as in [120], is common for objective relative evaluation assessment as it gives a more reliable quality estimate than standalone evaluation techniques [34].

3.4 Materials and Methods.

The purpose of this work is to investigate a method that is able to predict the per-case quality of a segmentation produced by any algorithm at large scale and within the clinical workflow. The method should not only give a prediction of the quality of the segmentation, but be able to identify if that segmentation has failed. To this end, this work employs RCA to evaluate the quality of individual segmentations in the absence of GT. Specifically, this work is a reimplementation and extensive validation of RCA applied to per-instance automated QC of 3D (2D-stack) CMR. It builds on the work in [164] which focused on multi-organ segmentation from whole-body MRI.

First, the dataset available through the UKBB imaging study is described then the method is outlined. Evaluation metrics used in this work are set out and the experiments are later explained.

3.4.0.1 The UK Biobank Imaging Study.

The UKBB [4, 108, 154] is a registered charity founded in 2006. Its aim is "to improve the prevention, diagnosis and treatment of a wide range of serious and life-threatening illnesses – including cancer, heart diseases, stroke, diabetes, arthritis, osteoporosis, eye disorders, depression and forms of dementia[1]".

In total, 500,000 participants were recruited to undergo health screening and donate blood, urine and saliva samples. Genomic data is also available. There is an aim for 100,000 participants to undergo an array of imaging procedures with multiple modalities on numerous anatomy. Many subjects repeat the assessments as part of a longitudinal study, with 20,000 Hospital Episode Statistics already recorded. All UKBB images are acquired at UKBB imaging centres in the UK. The UKBB resource is neither representative of the global or general UK populations. Research in [52] shows that the data have a 'healthy volunteer' selection bias.

Access to the UKBB data is open to all health researchers across the world conducting approved investigations for the public good. Applications must be approved and meet legal and ethical criteria. In this work, data was available under Applications 18545 and 2964. There is an expectation that any new derived measurements from UKBB data are submitted to the charity after their publication.

Part of the UKBB assessment process focuses on cardiac imaging including cine MRI, tagged MRI, T1-mapping and Phase-contrast imaging [123]. This work uses the T1-weighted CMR images from the study. The images are all stacks of 2D acquisitions with slice-thickness of 8.0 mm and slice-gap of 2 mm. The CMR scans have in-plane resolution of $[1.83 \times 1.83]$ mm and span around $[192,\ 208]$ pixels per slice. The number of slices per scan varies in the range $[4,\ 14]$ with an 89% majority having 9-12 slices.

[1] http://www.ukbiobank.ac.uk/

3.4.1 Reverse Classification Accuracy.

In RCA the idea is to build a classifier based solely on the *test* image using its predicted segmentation as *pseudo* GT. This classifier is then evaluated on a reference dataset for which segmentations are available. There are two possible outcomes to this procedure:

Case 1. Assuming that the predicted segmentation is of *good quality*, the created model should be able to segment at least one of the reference images with high accuracy. This is likely to be a reference image which is similar to the test image.

Case 2. If none of the reference images are segmented successfully, then the predicted segmentation is likely to be of *poor quality*.

These assumptions are valid if the reference dataset is representative of the test data. This is usually the case in the context of ML where the reference data could have been used in the first place to train the automated method for which the test performance is to be predicted. In this work, the training set and reference sets are distinct. Where training and test data differ, a drop in performance may be observed - this is explored in chapter 5. The advantage of RCA is that it detects whether there is a mismatch between training and test data. If the predicted segmentation is of poor quality, the RCA classifier should fail on all reference images. The performance of the RCA classifier on the reference set is measured with any chosen quality metric, *e.g.* DSC. The best score among all reference images determines the quality *estimate* for the predicted segmentation obtained for a test image.

The original work on RCA [164] explored a variety of possible classifiers that could be trained on a single test-image and its segmentation including Atlas Forests (AFs) [185] and CNNs. In this context, and throughout this work, an 'atlas' refers to an image-segmentation pair whose segmentation has been verified by a manual annotator. In [164], a simple single-atlas registration classifier outperformed both the AF and CNN approaches in predicting segmentation accuracy. For this reason, the single-atlas approach for the model in used in this work.

Registration is the process of aligning two or more images based upon similar content within them, *e.g.* structures or intensities, as described in section 2.4 on page 69. Rigid registration restricts the images to move only by linear translations and rotations. More complex non-rigid registration methods exists that allow for differences in scale between the images and for more complex distortions. The single-atlas registration classifier in RCA works by performing non-rigid registration of the test-image to a set of individual reference-images. The resulting transformations are used to warp the test-segmentation. This yields a set of warped segmentations which are quantitatively compared to the reference segmentations. The overlap between the pairs is calculated as the DSC whilst boundary agreement is computed using surface-distance metrics. The best metric values among the reference set are taken to be the prediction for the quality of the test-segmentation.

This work modifies the single-atlas registration approach of [164]. Processing and modifying the

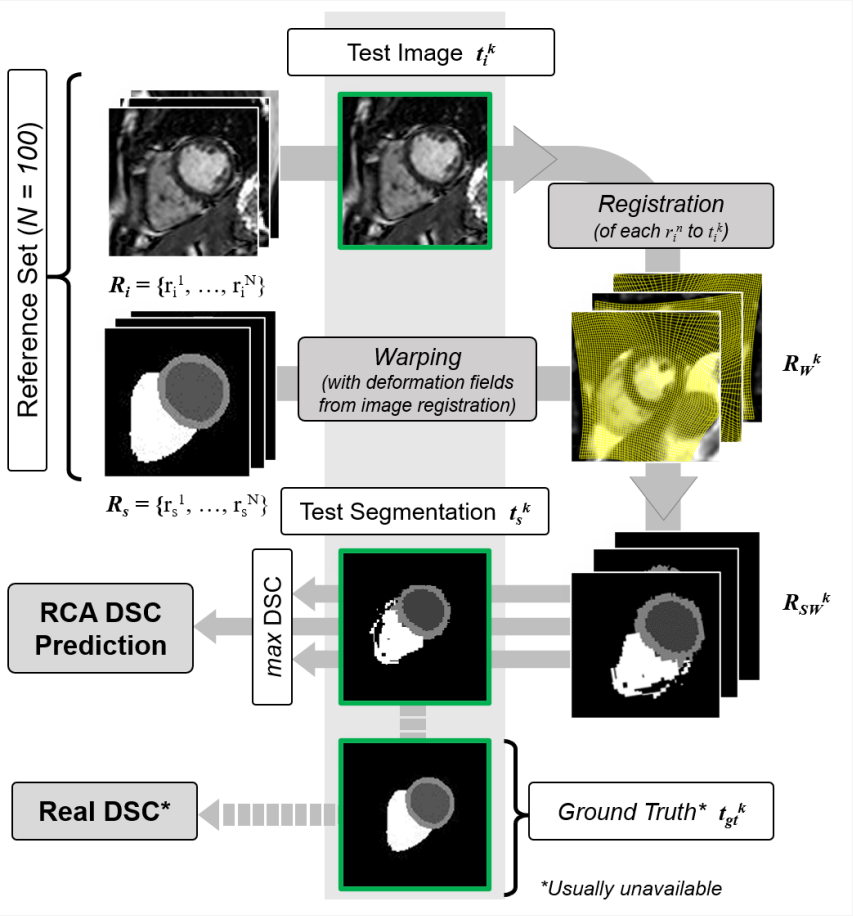


Figure 3.1: RCA Framework Using Single-atlas Registration Classifier. In practice, the GT of the test image is unavailable so 'real DSC' is unknown. Test image is segmented using the warped reference image GT. The best score obtained over all references is used as a proxy for determining the quality of the prediction on the test image.

test-segmentation is not usually desirable as this may introduce discretization artefacts adding False Positives (FPs) into the labelmap. Instead, the single-atlas registration is performed in reverse: the reference-images are registered to the test-image and the transformation used to warp the reference segmentations. This results in a set of warped segmentations in the test-image space which are then compared to the test-segmentation. Figure 3.1 gives an overview of RCA as applied in our study.

Formally, this work uses a reference set $\mathbf{R}_\mathrm{i} = \left\{r_\mathrm{i}^1, \cdots, r_\mathrm{i}^N\right\}$ of N cardiac atlases with reference segmentations $\mathbf{R}_\mathrm{s} = \left\{r_\mathrm{s}^1, \cdots, r_\mathrm{s}^N\right\}$. The test set $\mathbf{T}_\mathrm{i} = \left\{t_\mathrm{i}^1, \cdots, t_\mathrm{i}^M\right\}$ of M images has automatically generated predicted segmentations $\mathbf{T}_s = \left\{t_\mathrm{s}^1, \cdots, t_\mathrm{s}^M\right\}$ whose quality are to be to assessed. If the GT segmentations $\mathbf{T}_\mathrm{gt} = \left\{t_\mathrm{gt}^1, \cdots, t_\mathrm{gt}^M\right\}$ for $\mathbf{T}_\mathrm{i}$ exist, one can evaluate the accuracy of these quality assessments. Using RCA the quality of those predicted segmentations can be estimated and compared to the real quality with respect to the GT.

In the case where $m = k$, t_i^k is the k^th test image and its predicted segmentation is t_s^k. To apply

RCA, all reference images $\mathbf{R}_{\mathrm{i}}$ are first registered to t_{i}^{k} by performing a rigid registration in the form of a Center of Mass (CoM) alignment. Initial versions of this work [133] used landmark registration [116] at this stage, but CoM alignment reduces the computational cost. Non-linear registration is performed of each aligned reference image in $\mathbf{R}_{\mathrm{i}}$ to the test image to get warped reference images $\mathbf{R}_{\mathrm{iW}}^{k}$. The same transformations are used to warp the GT reference segmentations $\mathbf{R}_{\mathrm{s}}$ to get the set $\mathbf{R}_{\mathrm{sW}}^{k}$. Each warped segmentation in $\mathbf{R}_{\mathrm{sW}}^{k}$ is compared against the predicted segmentation t_{s}^{k} by evaluating a set of metrics. The best value for each metric over all warped reference segmentations is taken to be the *prediction* of segmentation accuracy for t_{s}^{k}. In these validation studies, the *real* metrics are computed by comparing the predicted segmentation t_{s}^{k} with its GT t_{gt}^{k}.

3.4.2 Evaluation of Predicted Accuracy

Assessing the quality of a segmentation requires some evaluation metrics. Many metrics have been proposed [157], but they can often have varying definitions and implementations. In this work, volumetric overlap and surface distances are considered as they are two distinct metrics indicating different aspects of segmentation quality.

The segmentation quality metrics predicted with RCA include the Dice Similarity Coefficient (DSC), Mean Surface Distance (MSD), Root Mean Squared Error (RMS) Surface Distance and Hausdorff Distance (HD). For each test image, evaluation metrics are reported for each class label: Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), and Right Ventricular Cavity (RVC). The voxels belonging to the papillary muscles are incorporated into the LVC class. The Right Ventricular Myocardium (RVM) is difficult to segment because it is very thin, therefore it is seldom seen in Short-Axis (SAX) CMR segmentations and is not considered in this work.

For each evaluation metric, DSC and surface distances, two different average values could be reported: either a whole-heart average by combining all class labels into a single Whole-Heart (WH) class or, second, by taking the mean across the individual class scores. The WH-class average is usually higher because a voxel attributed to an incorrect class will reduce the mean calculated across the classes, but will actually be considered correct in the single WH-class case.

3.4.2.1 Dice Similarity Coefficient.

For two segmentations, **A** and **B**, DSC is a measure of overlap given by equation (3.1) on the next page. It is equivalent to the F_1 score given in section 2.2.5 on page 51. In the calculation for DSC, the order of **A** and **B** does not matter; the DSC is a symmetric measure. If some structure in an image has been 'under' or 'over'-segmented by the same amount, the DSC will be equal. In [126], the authors discuss this as a 'sensitivity-specificity' trade-off where one is usually more important in most clinical settings. They propose a 'coverage-factor' metric based on a discrepancy measure. Whilst this metric has clear advantages, it is not so widely used as the DSC which will be considered in this work.

Dice Similarity Coefficient:
$$\mathrm{DSC} = \frac{2\,|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A}| + |\mathbf{B}|} \tag{3.1}$$

3.4.2.2 Surface Distances.

The surface distance $\mathrm{SD}_{a,B}$ between a point a on the surface A and the surface B is given by the minimum of the euclidean norm for all points b in the surface B given in equation (3.2). The norm or absolute value is usually taken to account for points where surfaces may intersect. It finds the closest point on B to a. The total surface distance SD_{AB} between A and B is the sum of the surface distances for all points in A given in equation (3.3). We don't assume symmetry in the calculations, so the surface distance SD_{BA} is also calculated from B to A.

Surface Distance:
$$\mathrm{SD}_{a,B} = \min_{b \in B} ||a - b||_2 \tag{3.2}$$

Total Surface Distance:
$$\mathrm{SD}_{AB} = \sum_{a \in A} \mathrm{SD}_{a,B} \tag{3.3}$$

Mean Surface Distance:
$$\mathrm{MSD}_{AB} = \frac{1}{N_A} \sum_{a \in A} \mathrm{SD}_{a,B} \tag{3.4}$$

By taking the mean over all N_A points, *i.e.* pixels in the boundary, in A the Mean Surface Distance (MSD) in equation (3.4) is found. The Root Mean Squared Error (RMS) surface distance is calculated by taking the square of surface distances, averaging and taking the square root. Finally, the Hausdorff Distance (HD) is the maximum of the surface distances calculated between the two surfaces as shown in equation (3.6). Again, this quantity is not necessarily symmetric. A visual representation of the surface distances is shown in figure 3.2 on the next page.

RMS Surface Distance:
$$\mathrm{RMS}_{AB} = \sqrt{\frac{1}{N_A} \left(\sum_{a \in A} \mathrm{SD}_{a,B}^2 \right)} \tag{3.5}$$

Hausdorf Distance:
$$\mathrm{HD}_{AB} = \max_{a \in A} \mathrm{SD}_{a,B} \tag{3.6}$$

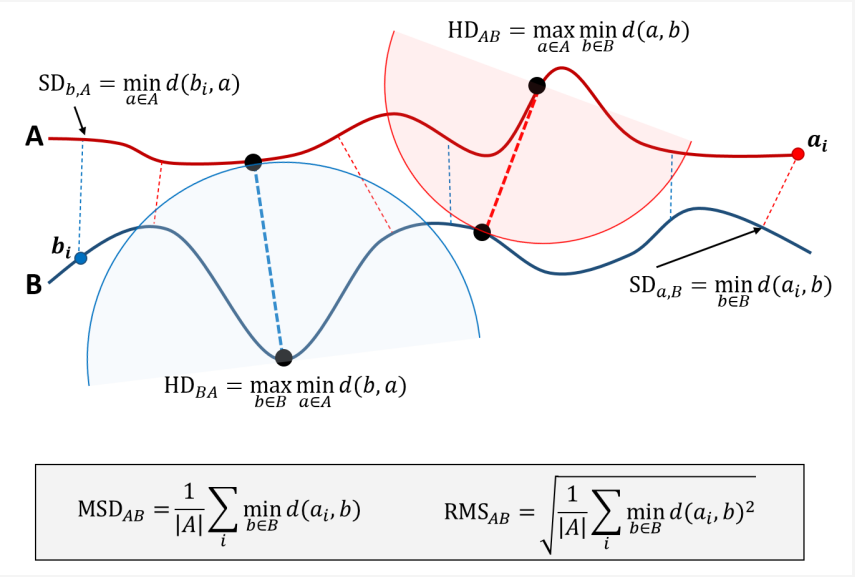


Figure 3.2: Visualization of Surface Distances. The point a_i on line A is separated from the nearest point on line B by $SD_{a_i,B} = \min_{b\in B} d(a_i,b)$ where $d(.\,,.)$ is some distance measure. The total surface distance between A and B is $S_{AB} = \sum_i SD_{a_i,B}$. The operation is not necessarily symmetric such that $SD_{AB} \neq SD_{BA}$. Hausdorff Distance (HD) is the maximum surface distance, Mean Surface Distance (MSD) and RMS are shown.

3.4.3 Experimental Setup.

Three investigations are conducted in this work which are summarised in table 3.1:

A An initial small-scale validation study on 400 segmentations of 80 images from an internal cardiac atlas dataset.

B A large-scale validation study on a large set of 4,805 UKBB images with manual GT provided by Hammersmith Hospital, London.

C A real-world application to a further set of 7,250 UKBB 3D CMR segmentations forming part of a Genome-wide Association Study (GWAS).

Table 3.1: A summary of the RCA experiments performed in this study. Experiment A uses data from an internal dataset which is segmented with a multi-atlas segmentation approach and manually validated by experts at Hammersmith Hospital, London. Manual validations are counted as GT and 100 of them are taken for the reference set used in all experiments. UKBB datasets are shown with their respective application numbers. In experiment C segmentations are performed with both a RFs and a CNN. In C the CNN from [9] is used.

Experiment	Dataset	Size	GT	Seg. Method
A	Hammersmith	100	Yes	RF
B	UKBB-2964	4,805	Yes	RF and CNN
C	UKBB-18545	7,250	No	Multi-Atlas

3.4.3.1 Reference Dataset.

The reference image set is the same in all investigations. The set comprises 100 3D (2D-stack) SAX End-Diastolic (ED) CMR scans that were automatically segmented and then validated by expert clinicians at Hammersmith Hospital, London. This reference set is distinct from all other datasets used. Compared with data from the UKBB, the reference set is of higher in-plane resolution at $[1.25 \times 1.25]$ mm and have a smaller slice thickness of 2 mm. These images are not used for any purpose other than for this reference set. Throughout, this reference set is referred to as $\mathcal{R}$.

When choosing a reference set, one should ensure that it is representative of the dataset on which it is being used: it should be of the same domain (SAX CMR in this case). It should also be large enough to capture some variability across the dataset. A reference set that is too small may underestimate the RCA prediction. Though it can be argued that this may be better than overestimating the quality of a segmentation. Conversely, too large a reference set will cause a significant lengthening of RCA execution time. The effect of the RCA reference set size on prediction accuracy is discussed in section 3.6 on page 115.

3.4.3.2 Experiment A: Initial Validation Study.

Data: RCA is validated on predicting cardiac image segmentation quality using a set $\mathcal{T}$ of 100 manually verified image-segmentation pairs from the Hammersmith dataset. $\mathcal{T}$ is distinct from the reference dataset $\mathcal{R}$. Each pair in $\mathcal{T}$ contains a SAX ED 3D (2D-stack) CMR and its manual segmentation. The images have a pixel-resolution of $[1.25 \times 1.25 \times 2.0]$ mm and span $[256,\ 256,\ 56]$ voxels. Each manual segmentation identifies voxels belonging to the LVC, LVM and RVC separating the heart from the background class.

For validation, automatic segmentations of each atlas image is generated. A RF model is employed with $T = 500$ trees and a maximum depth of $D = 40$. The model is trained on the 100 image-segmentation pairs in $\mathcal{T}$. RFs can produce a variety of test segmentations with intentionally degraded segmentation quality by limiting the depth of the trees during test time. The trained RF is used to generate 4 segmentations for each of the 100 images in $\mathcal{T}$ by using depths of 5, 20, 30 and 40 during evaluation. Thus, a total of 400 segmentations are used in this initial validation study.

Evaluation: RCA is performed on all 400 segmentations to yield predictions of segmentation quality. The manual GT segmentations are used to evaluate the real metrics for each automated segmentation. These values are compared to the quality predicted by RCA. To identify individual cases where segmentation has failed, a simple classification strategy is implemented similar to that in [164]. A 2-group binary classification scheme is set up where DSC in the range $[0.0,\ 0.7)$ are considered 'poor' and in the range $[0.7,\ 1.0]$ are 'good'. These boundaries are chosen arbitrarily for this task and would be adjusted for a particular use-case when deployed. For example, it could be that much higher DSCs are required in neuroimaging or where segmentations are used in volume estimation tasks. Other RCA evaluation strategies could be employed on a task-specific basis, *e.g.* formulating an outlier detection problem with different statistical measures. The thresholding approach chosen in this validation study allows the formulation of a confusion matrix from which True Positive Rate (TPR), False Positive Rate (FPR) and an overall accuracy can be calculated.

3.4.3.3 Experiment B: Validation on Large-scale Population Imaging Data

This experiment demonstrates that RCA is robust for deployment in large-scale studies, and produces accurate predictions of segmentation quality on an individual basis. This work was conducted in collaboration with a consortium of colleagues under UKBB Application 2964.

Data: The CMR scans described in section 3.4.0.1 on page 96 are used in this work. Images are not cropped or resampled: this allows RCA to use more information during the registration process.

Petersen and colleagues [122, 124] manually segmented all slices of the 3D CMR scans available under the data access application. Several annotators were employed following a standard op-

erating procedure and generated almost 4,805 high-quality segmentations. With these manual segmentations acting as GT they can be directly compared with predicted labelmaps to evaluate real quality metrics at large scale. It is likely that most of the 4,805 segmentations are of good quality. New automated segmentations are generated for each image using the same RF trained in section 3.4.3.2 on the previous page. Each image is segmented using the forest evaluated at a random tree depth yielding segmentations of varying quality.

In addition to the RF segmentations, RCA is evaluated with 900 segmentations generated with a recent DL-based approach. As part of the same UKBB Application 2964, Bai *et al.* [9] trained a CNN on 3,900 manually segmented images. The remaining 900 were then automatically segmented using the trained network. The results of this CNN approach reflects the state-of-the-art in automated 3D CMR segmentation with an accuracy matching the performance human experts.

Evaluation: RCA is conducted on all 4,805 RF segmentations to yield predictions of segmentation quality. RCA is also performed separately on the 900 CNN segmentations produced by the state-of-the-art deep learning approach. With the availability of GT manual segmentations, this experiment can be evaluated in the same way as Experiment A in section 3.4.3.2 on the preceding page.

3.4.3.4 Experiment C: Automatic vs. Manual QC at Large-scale.

Having evaluated RCA in experiments A and B, this experiment mimics how the approach would behave in a real-world application where the GT is unavailable. RCA is again applied to segmentations of UKBB images but in the case where no manual contours have been provided.

Data: In total, 7,250 CMR images are available UKBB Application 18545. Each image has been automatically segmented using a multi-atlas segmentation approach [8]. As part of a GWAS, each automatic segmentation has been checked manually to confirm segmentation quality. There are no GT segmentations for these images, but the segmentations are assessed by a clinical expert. The manual QC is based only on visual inspection of the basal, mid and apical layers of the image. For each layer a score between 0 and 2 is assigned based on the quality of *only* the LVM segmentation. The total QC score is thus in the range $[0, 6]$, where 6 would be considered as a highly accurate segmentation. Scores for individual layers were not recorded. Where the UKBB images have a poor Field-of-view (FOV), the segmentations were immediately discarded for use in the GWAS study: in this work, these images are given a score of -1. For the GWAS study, 'poor FOV' indicates any image in which the *entire* heart was not visible. It should be expected that despite the poor FOV of these images, the labelmaps themselves may still be of good quality as the automated segmentation algorithms can still see most of the heart.

Out of the 7,250 segmented images, 152 have a 'bad FOV' (QC $= -1$) and 42 have an obviously poor segmentation (QC $= 0$). There are 2, 14, 44, 300, 2866 and 3830 images having QC scores 1 to 6 respectively. This investigation explores how well RCA-based quality predictions correlate

with the manual QC scores.

Evaluation: RCA is performed on all 7,250 segmentations to yield predictions of segmentation quality for the LVM. With the absence of GT segmentations, the evaluation conducted in experiments A and B cannot be performed. In this case, the correlation is calculated between the predicted scores from RCA (for LVM) and the manual QC scores. A visual inspection of individual cases is also performed within the QC categories.

3.5 Results.

When RCA is run on a predicted segmentation, 100 registrations are performed: one for each reference image to the test image. Subsequently, each deformation field learned during the registration process is used to transform the corresponding reference segmentation to test-image space, thus 100 warpings are performed. 4 metrics are calculated per reference image and the final RCA quality prediction given by the largest calculated DSC. An example of the output from RCA is shown in figure 3.3.

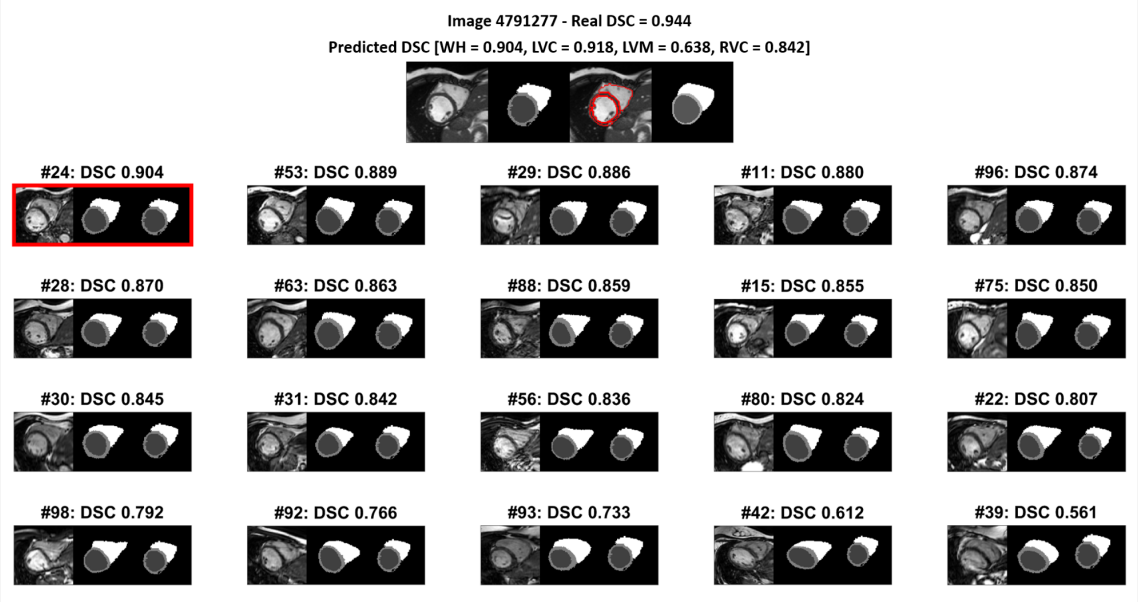


Figure 3.3: Example Result from RCA Analysis. Examples of an RCA result on one predicted segmentation. The panel in the top row shows (left to right) the MRI scan, predicted segmentation, an overlay, and the manual annotation. The array below shows a subset of the 100 reference images $\mathcal{R}$ ordered by DSC at regular intervals. Number above each panel is the index of the reference image. Each panel in the array shows (left) the reference image, (middle) its GT segmentation and (right) the test-segmentation - from the upper panel - which has been warped to the reference image. The real DSC between each reference image and warped segmentation is shown for each pair. Reference image in red indicates the one chosen for RCA prediction, i.e. maximum DSC. RCA-predicted and real GT-calculated DSCs are shown for the WH binary classification case at the top alongside the metrics for each individual class in the segmentation: LVC, LVM and RVC.

Table 3.2: Results of Initial RCA Validation on 400 RF Segmentations. Reverse Classification Accuracy (RCA) predictions on 400 Cardiac Magnetic Resonance Imaging (CMR) segmentations generated with Random Forest (RF). Classes are Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), Right Ventricular Cavity (RVC), an average over the classes (Av.) and a binary segmentation of the Whole-Heart (WH). (left) First row for each class shows the binary classification accuracy for 'poor' and 'good' segmentations in the DSC ranges $[0.0\ 0.7)$ and $[0.7\ 1.0]$ respectively. Second row shows the binary classification accuracy for 'poor' and 'good' segmentations in the MSD ranges $[> 2.0]$ mm and $[0.0,\ 2.0]$ mm respectively. TPRs and FPRs are also shown. (right) Mean Absolute Error (MAE) is reported on the predictions of DSC and additional surface-distance metrics: Mean Surface Distance (MSD), Root Mean Squared Error (RMS) and Hausdorff Distance (HD).

Class	Metric	Acc.	TPR	FPR
LVC	DSC	0.973	0.977	0.036
	MSD	0.980	0.975	0.019
LVM	DSC	0.815	0.947	0.215
	MSD	0.990	0.987	0.008
RVC	DSC	0.985	0.923	0.012
	MSD	0.943	0.914	0.047
Av.	DSC	0.924	0.949	0.089
	MSD	0.971	0.959	0.025
WH	DSC	**0.988**	**0.979**	**0.000**
	MSD	**0.948**	**0.886**	**0.048**

	MAE			
Class	**DSC**	**MSD** (mm)	**RMS** (mm)	**HD** (mm)
LVC	0.020	4.104	5.593	14.15
LVM	0.044	3.756	4.741	13.08
RVC	0.030	4.104	5.022	16.63
Av.	0.031	3.988	5.119	14.62
WH	**0.029**	**4.445**	**5.504**	**15.11**

3.5.1 Experiment A Results: CMR Atlases with Verified GT.

The 100 atlas images, distinct from the reference set $\mathcal{R}$, were re-segmented with the RF classifier at different depths to generate 400 segmentations of varying quality. RCA was performed on the 400 segmentations to assess their quality. A summary of the results is shown in table 3.2.

There is low MAE across all evaluation metrics when the Whole-Heart (WH) is considered. The scatter plots in figure 3.4 on the next page on real and predicted scores illustrate the very good performance of RCA in predicting segmentation quality scores. It is also found that from the 400 test segmentations, RCA is able to classify 'good' ($\mathrm{DSC} \in [0.7,\ 1.0]$) and 'poor' ($\mathrm{DSC} \in [0.1,\ 0.7)$) segmentations with an accuracy of 99%. From 171 poor segmentations at this DSC threshold, 166 (97.1%) could be correctly identified by RCA. 100% of good-quality segmentations were correctly labelled.

Additionally, binary classification accuracy of 95% is seen when applying a threshold of 2.0 mm on the MSD. From 365 poor segmentations at this threshold, 348 (95.3%) could be correctly identified by RCA. Similarly, 31 from 35 (88.6%) good-quality segmentations were correctly labelled.

For all evaluation metrics, there is a strong, positive linear relationship between predicted and real

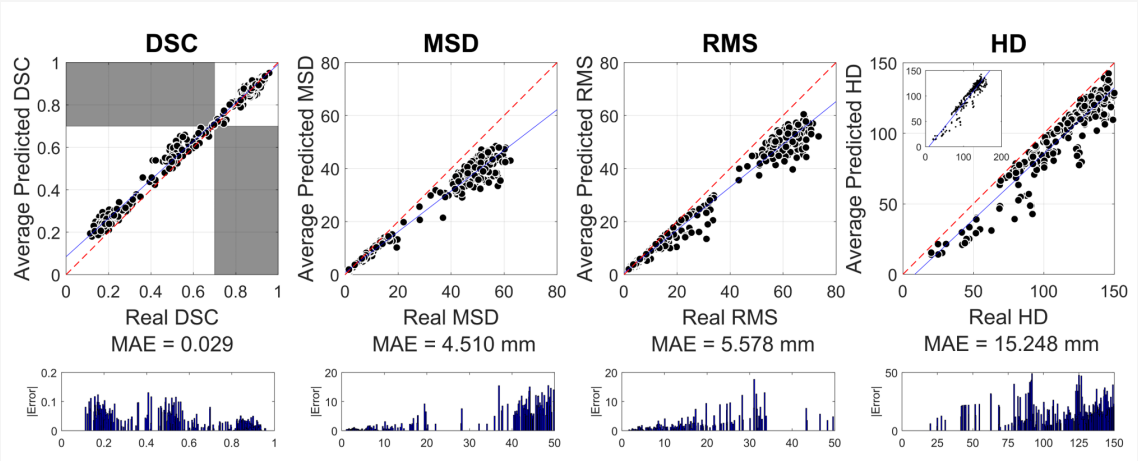


Figure 3.4: RCA Validation on 400 RF Segmentations with manual GT - Whole Heart Class. RCA with single-atlas classifier was used to predict (left-to-right) DSC, MSD, RMS and HD for 400 CMR segmentations generated with a Random Forest (RF) classifier. Real metrics calculated on available Ground Truth (GT) labelmaps. All calculations on the WH binary classification task. Low MAE for all metrics and 99% binary classification accuracy (TPR $= 0.98$, FPR $= 0.00$) with a DSC threshold of 0.70. High accuracy for individual segmentation classes. Absolute error for each image is shown for each metric. There increasing error with decreasing quality of segmentation based on real metric score.

values with $r \in [0.95, 0.99]$ and $p < 0.0001$. Further analysis of the data shows increasing MAE for each metric as the real score gets worse, *e.g.* the error for the predicted MSD increases with increasing surface distance. This correlates larger MAE with lower segmentation quality. In addition, when only those segmentations where the real metric is 30 mm or less are considered, the MAE drops significantly to 0.65, 1.71 and 6.78 mm for MSD, RMS and HD respectively. Greater errors for poor segmentations are not concerning, as they are still likely to be identified as having failed by RCA.

These patterns can be seen in the individual segmentation classes for the different cardiac regions. Figure 3.5 on the following page shows similar scatter plots for the LVC, LVM and RVC.

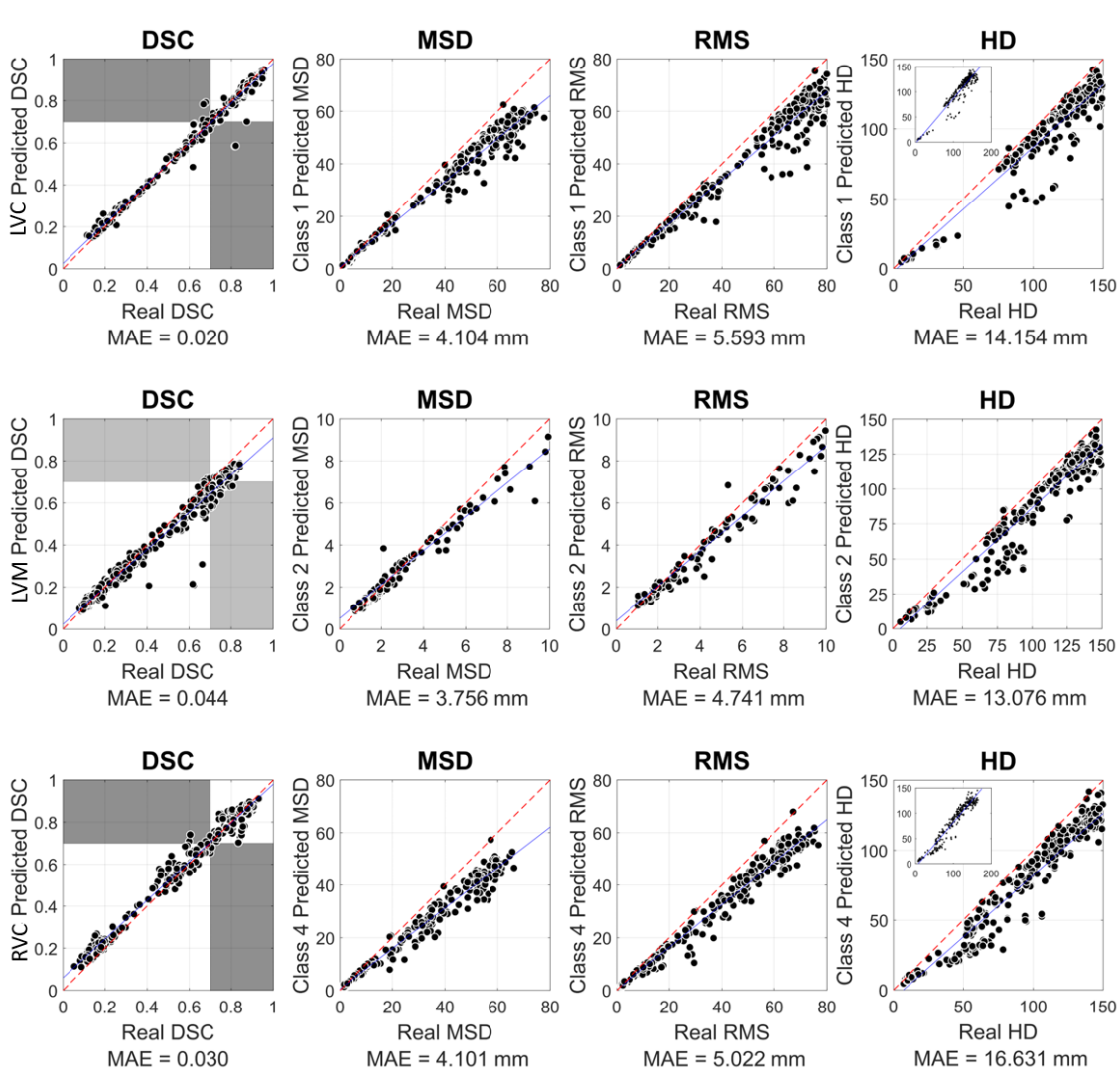


Figure 3.5: RCA Validation on 400 RF Segmentations with manual GT - Individual Classes. Reverse Classification Accuracy (RCA) with single-atlas classifier was used to predict (left-to-right) DSC, MSD, RMS and HD for 400 CMR segmentations generated with a Random Forest (RF) classifier. Real metrics calculated on available Ground Truth (GT) labelmaps. Low MAE for all metrics. High accuracy for individual segmentation classes (top-to-bottom) LVC, LVM and RVC.

Table 3.3: Results of RCA Validation on 4,805 RF Segmentations from UKBB. Classes are Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), Right Ventricular Cavity (RVC), an average over the classes (Av.) and a binary segmentation of the Whole-Heart (WH). (left) First row for each class shows the binary classification accuracy for 'poor' and 'good' segmentations in the DSC ranges $[0.0\ 0.7)$ and $[0.7\ 1.0]$ respectively. Second row shows the binary classification accuracy for 'poor' and 'good' segmentations in the MSD ranges $[> 2.0\text{mm}]$ and $[0.0\text{mm}\ 2.0\text{mm}]$ respectively. TPRs and FPRs are also shown. (right) MAE is reported on the predictions of DSC and additional surface-distance metrics: Mean Surface Distance (MSD), Root Mean Squared Error (RMS) and Hausdorff Distance (HD).

Class	Metric	Acc.	TPR	FPR
LVC	DSC	0.968	0.997	0.330
	MSD	0.975	0.962	0.011
LVM	DSC	0.454	0.956	0.571
	MSD	0.972	0.962	0.012
RVC	DSC	0.868	0.957	0.352
	MSD	0.969	0.977	0.040
Av.	DSC	0.763	0.970	0.418
	MSD	0.972	0.967	0.032
WH	DSC	**0.954**	**0.966**	**0.148**
	MSD	**0.978**	**0.984**	**0.027**

	MAE			
Class	**DSC**	**MSD** (mm)	**RMS** (mm)	**HD** (mm)
LVC	0.042	0.906	2.514	11.09
LVM	0.125	0.963	2.141	11.83
RVC	0.057	1.140	2.790	15.23
Av.	0.075	1.003	2.482	12.72
WH	**0.035**	**1.156**	**2.762**	**12.52**

3.5.2 Experiment B Results: UKBB CMR with Manual GT.

Results for the RF segmentations are shown in table 3.3. Binary classification accuracy of 95% is reported with a DSC threshold of 0.7. Low MAE is also achieved on the DSC. From 589 poor segmentations at this threshold, 443 (75.2%) could be correctly identified by RCA. Similarly, 4139 from 4216 (98.2%) good-quality segmentations were correctly labelled. Additionally, binary classification accuracy of 98% is achieved when applying a threshold of 2.0 mm on the MSD. From 2497 poor segmentations at this threshold, 2429 (97.3%) could be correctly identified by RCA. Similarly, 2270 from 2308 (98.3%) good-quality segmentations were correctly labelled.

The TPRs are high across the classes, showing that RCA is able to correctly and consistently identify 'good' quality segmentations. MSD-based FPRs are shown to be lower than those based on DSC, this would indicate that MSD is more discriminative for 'poor' quality segmentations and does not mis-classify them so much as DSC. Only two instances are identified where RCA predictions do not conform to the overall trend and predict much higher than the real DSC. On inspection, it is seen that the GT of these segmentations were missing mid-slices causing the real DSC to drop. These points can be seen in the upper-left-hand quadrant on figure 3.6 on the following page. The figure also shows that, over all metrics, there is high correlation between predicted and real quality metrics. This is very much comparable to the results from the initial validation study, Experiment A in section 3.5.1 on page 107. The strong relationship between the predicted quality metrics from

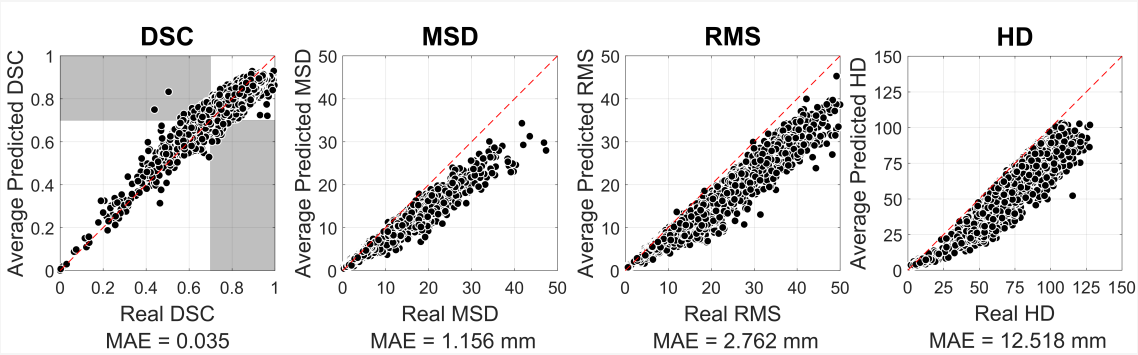


Figure 3.6: RCA Validation on 4,805 RF Segmentations with manual GT. 4,805 CMR were segmented with a Random Forest (RF) classifier. Manual Ground Truth (GT) contours available through UKBB Application 2964. RCA with single-atlas classifier used to predict DSC, MSD, RMS and HD. All calculations on the WH binary classification task. Low MAE reported for all metrics and 95% binary classification accuracy (TPR $= 0.97$ and FPR $= 0.15$) with DSC threshold of 0.70. High accuracy for individual segmentation classes: LVC, LVM and RVC.

RCA and the equivalent scores calculated with respect to the manual segmentations demonstrates concretely that RCA is capable of correctly identifying, on a case-by-case basis, segmentations of poor quality in large-scale imaging studies.

On the segmentations generated with the state-of-the-art human-level CNN described in [9], a 99.8% accuracy is reported for binary classification on the WH class. With a DSC threshold set at 0.7, RCA correctly identified 898 from 900 (99.8%) good-quality segmentations with only 2 false-negatives. A visualization of this can be seen in the top panel of figure 3.7 on the following page where the predicted and real DSC can be seen clustered in the high-quality corner of each metric's plot, *i.e.*, upper-right for DSC and lower-left for surface-distance metrics. This reflects the high quality segmentations of the DL approach which have been correctly identified using RCA. Table 3.4 on the next page shows the detailed statistics for this experiment.

The individual class accuracy for the LVM is lower in the CNN case when DSC is used as the quality metric. The results for this class are shown in the bottom panel of figure 3.7 on the following page. Segmentors can have difficulty with this class due to its more complex shape. From the plotted points, all cases can be seen to fall into a similar cluster to the average WH case, but the RCA score under-predicts the real DSC. This exemplifies a task-specific setting for how RCA would be used in practice. In this case one cannot rely only on DSC to predict the quality of the segmentation, so MSD could provide a more appropriate quality prediction.

Table 3.4: Results of RCA Validation on 900 CNN Segmentations from UKBB. Classes are Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), Right Ventricular Cavity (RVC), an average over the classes (Av.) and a binary segmentation of the Whole-Heart (WH). (left) First row for each class shows the binary classification accuracy for 'poor' and 'good' segmentations in the DSC ranges $[0.0\ 0.7)$ and $[0.7\ 1.0]$ respectively. Second row shows the binary classification accuracy for 'poor' and 'good' segmentations in the MSD ranges $[> 2.0\text{mm}]$ and $[0.0\text{mm}\ 2.0\text{mm}]$ respectively. TPRs and FPRs are also shown. (right) MAE is reported on the predictions of DSC and additional surface-distance metrics: Mean Surface Distance (MSD), Root Mean Squared Error (RMS) and Hausdorff Distance (HD).

Class	Metric	Acc.	TPR	FPR
LVC	DSC	0.998	1.000	0.000
	MSD	1.000	1.000	0.000
LVM	DSC	0.051	1.000	0.001
	MSD	1.000	1.000	0.000
RVC	DSC	0.901	1.000	0.033
	MSD	0.997	0.997	0.000
Av.	DSC	0.650	1.000	0.011
	MSD	0.999	0.999	0.000
WH	DSC	**0.998**	**1.000**	**0.000**
	MSD	**1.000**	**1.000**	**0.000**

	MAE			
Class	**DSC**	**MSD** (mm)	**RMS** (mm)	**HD** (mm)
LVC	0.082	0.386	0.442	1.344
LVM	0.268	0.510	0.547	2.127
RVC	0.146	0.588	0.656	2.086
Av.	0.165	0.495	0.548	1.852
WH	**0.089**	**0.460**	**0.509**	**1.698**

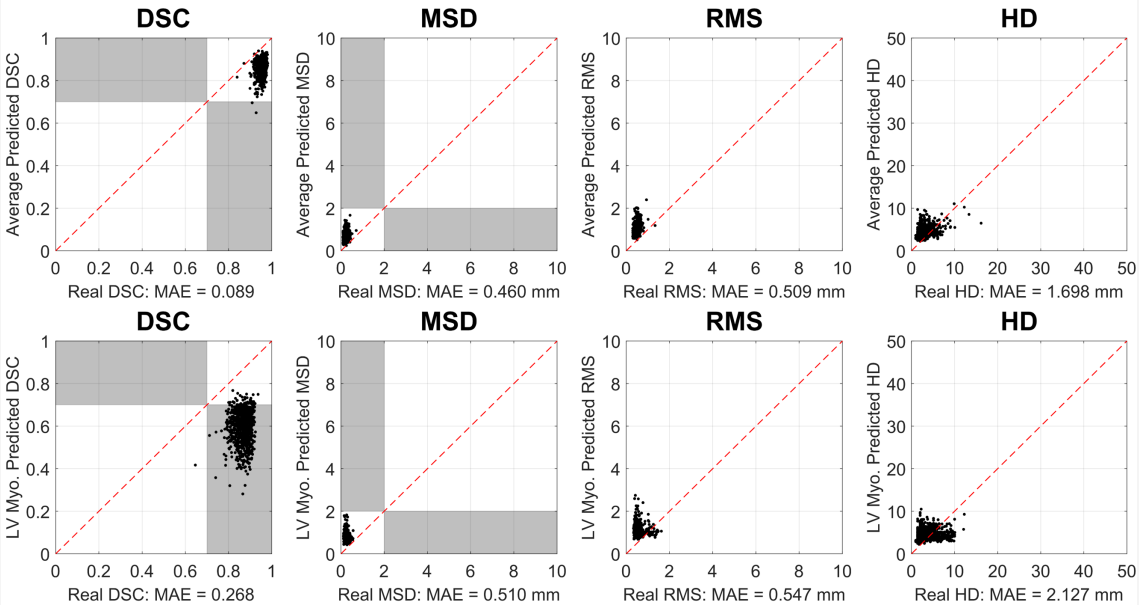


Figure 3.7: RCA Validation on 900 CNN Segmentations with manual GT. CNN segmentation conducted as in Bai *et al.* [9]. Manual contours available through UKBB Application 2964. RCA with single-atlas classifier used to predict DSC, MSD, RMS and HD. All calculations for the binary quality classification task on (top) WH average and (bottom) LVM. Low MAE for all metrics and 99.8% binary classification accuracy (TPR $= 1.00$ and FPR $= 0.00$) with DSC threshold 0.70.

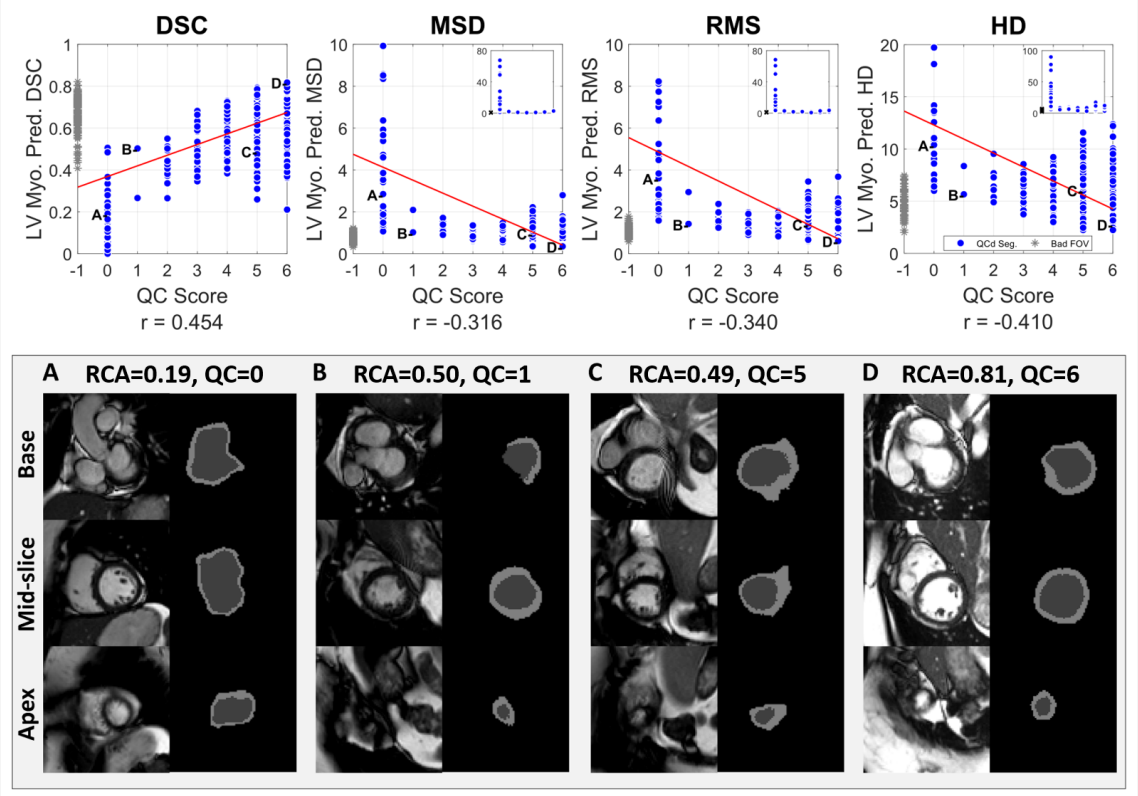


Figure 3.8: RCA Application to 7,250 RF segmentations with manual QC Scores. 7,250 CMR segmentations generated with a multi-atlas segmentation approach [8]. Manual QC scores given in the range $[0\,6]$ (i.e. $[0\,2]$ for each of basal, mid and apical slices). RCA with single-atlas classifier used to predict DSC, MSD, RMS and HD. All calculations on LVM binary classification task. Correlation is shown in all metrics. Examples show: A) and B) agreement between low predicted DSC and low manual QC score, C) successful automated identification of poor segmentation with low predicted DSC despite high manual QC score and D) agreement between high predicted DSC and high manual QC score. Inserts in top row display extended range of y-axis.

3.5.3 Experiment C Results: UKBB CMR without GT

Figure 3.8 shows the relationship between manual QC scores and the predicted DSC, MSD, RMS and HD obtained from RCA. These predictions are for the LVM *only* and not the overall segmentation as this class was the focus of the manual QC procedure. Manual QC was not performed for the other classes.

Figure 3.8 also shows a sample of segmentations with manual QC scores of 0, 1, 5 and 6 for the LVM. With a score of 0, example 'A' must have a 'poor' quality segmentation of LVM at the basal, apical and mid slices. Example 'B' shows relatively low surface-distance metrics and a low DSC, we see this visually as the boundary of the myocardium is in the expected region, but is incomplete in some slices. This segmentation has been given a score of 1 because the mid-slice is well segmented while the rest is not; which is correctly identified by RCA. In example 'C', the

segmentation of the LVM is clearly not good with respect to the image, yet it has been given a manual QC score of 5. Again, RCA is able to pick up such outliers by predicting a lower DSC. The final example 'D' displays an agreement between the high predicted DSC from RCA and the high manual QC score. These examples demonstrate RCA's ability to correctly identify both good and poor quality segmentations when performing assessments over an entire 3D segmentation. It also demonstrates the limitations of manual QC and the success of RCA in identifying segmentation failure on a per-case basis.

3.6 Conclusions.

A RCA can predict the quality of CMR segmentations on a per-case basis.

Results show that it is possible to assess the quality of individual CMR segmentations at large-scale and in the absence of ground truth. Previous approaches have primarily focused on evaluating the overall, average performance of segmentation methods or required large sets of pre-annotated data of good and bad quality segmentations for training a classifier. The RCA method is well suited for use in image-analysis pipelines and clinical workflows where the quality of segmentations should be assessed on a per-case basis.

B RCA can be preferable over rudimentary, manual QC scoring.

Creating a set of manual QC scores for over 7,200 images is a laborious task but it has provided worthwhile evidence for the utility of RCA on large-scale studies. It is clear, however, that the 3-layer inspection approach with a single-rater has limitations. First, inspecting all layers would be preferable, but it is highly time-consuming and, second, with multiple raters, averaging or majority voting could be employed to reduce human error.

RCA is unlikely to mimic the exact visual manual QC process, and it should not, as it naturally provides a different, more comprehensive, assessment of segmentation quality. The manual QC approach is a relatively crude assessment of segmentation quality, and so a direct, quantitative comparison using the visual QC categories was not performed. Rather, it has been demonstrated that there is a general correlation between manual QC and the predicted RCA score.

C RCA run-time per prediction is too long for applications with real-time constraints.

The RCA validation process was carried out on 8-core Intel i7 3.6 GHz machines. The whole process for a single test segmentation - including 100 reference image registrations, warping 100 reference segmentations and metric evaluations - took 7-10 minutes, making it suitable for background processing in large-scale studies and clinical practice. However, this is a limitation as the run-time per case currently does not allow immediate feedback and prohibits applications with real-time constraints. For example, one could envision a process where CMR scans are immediately segmented after acquisition, and feedback on the quality would be required while the patient is still in the scanner. For this, the computation time of RCA would need to be reduced possibly through an automatic selection of a subset of reference images, which is discussed below. To address the run-time issue, this work explores a Deep Learning (DL)-based RCA framework in chapter 4. Preliminary results are also published in [132]. With a real-time RCA framework, the method could be used to identify challenging cases for CNN-based segmentors where the RCA feedback could be used to improve the initial segmentation algorithm.

D The size of the RCA reference set $\mathcal{R}$ is crucial.

It has already been made clear that the reference images should be from the same domain as the intended targets. However, the size of the reference set should also be considered. Using a subset of $\mathcal{R}$ could help to optimize the run-time of RCA predictions.

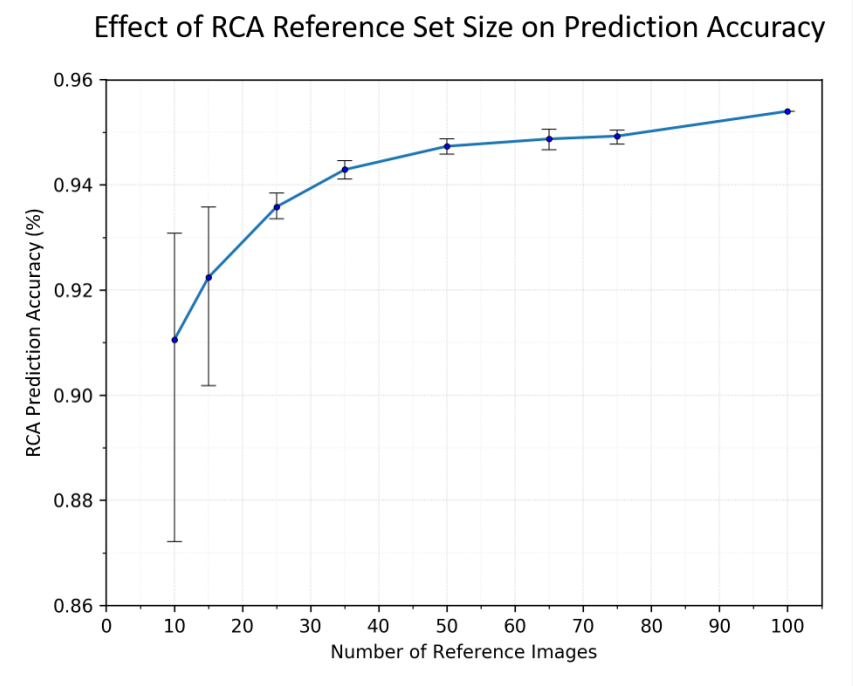
To better understand the effect of reference set size on prediction accuracy, an empirical evaluation was performed using the data from Experiment B in section 3.4.3.3 on page 103. Five different randomly selected subsets of the reference set $\mathcal{R}$ were created containing sizes 10, 15, 25, 35, 50, 65 and 75 images-segmentation pairs. Each set was used for obtaining RCA predictions on the 4,805 segmentations from Experiment B. Figure 3.9 on the following page shows the mean accuracy computed across 5 runs for each reference set size with a random set of references chosen in each run. Error bars indicate the highest and lowest accuracy achieved across the five runs. Accuracy is computed using the same DSC threshold of 0.7 as used in Experiment B.

The figure shows that the mean accuracy increases with increasing number of reference images. The error bars in Figure 3.9 on the next page show a decrease in size with increasing size of the reference set. As the reference set grows in size, a greater variability in the images is captured that allows the RCA process to become more accurate. It is interesting that even with small reference sets of about 20 images, high accuracy of more then 90% can be obtained.

E RCA does not give a one-to-one mapping between real and predicted evaluation metrics.

Although RCA can give a good indication of the real DSC score for an individual segmentation, an accurate one-to-one mapping between the predicted and real DSC has not been achieved. This is also true for the surface-distance metrics. However, it has been shown that the method will confidently differentiate between 'good' and 'poor' quality segmentations based on an application-specific threshold. The threshold could be chosen depending on the application's requirements for what qualifies as a 'good' segmentation. Failed segmentations could be re-segmented with different parameters, regenerated with alternative methods, discarded from further analyses or, more likely, sent to a user for manual inspection. Additionally, whilst RCA has been shown to be robust to cardiovascular topology it would need to be re-evaluated for use in other anatomical regions.

Figure 3.9: Effect of RCA Reference Set Size on RCA Prediction Accuracy. 4,805 automated segmentations from Experiment B in section 3.4.3.3 on page 103 were processed with RCA using differing numbers of reference images. Random subsets of reference images were taken from the full set of 100 available reference images in $\mathcal{R}$ to create subsets $\mathcal{R}_i, i \in \{10, 15, 35, 50, 65, 75\}$. Five runs were performed with a different random selection of reference images. Error bars show variation across the runs. Average prediction accuracy increases with increasing number of reference images and the variance between runs also decreases.



Chapter 4

A Deep Learning-based framework for Real-time Segmentation Quality Control in Cardiac Magnetic Resonance Imaging

Recent advances in Deep Learning (DL)-based image segmentation methods have enabled real-time performance with human-level accuracy. However, even the best method occasionally fails due to low image quality, artifacts or unexpected behaviour of black box algorithms. Being able to predict segmentation quality in the absence of Ground Truth (GT) is of paramount importance in clinical practice, but also in large-scale studies to avoid the inclusion of invalid data in subsequent analysis. Additionally, there may be scenarios where immediate feedback on the quality of a segmentation is desirable or rapid, high-throughput analysis is required; thus a real-time framework is needed.

In this work, two approaches are proposed for real-time automated Quality Control (QC) for Cardiac Magnetic Resonance Imaging (CMR) segmentations using DL. First, a Convolutional Neural Network (CNN) is trained on 12,880 samples to predict Dice Similarity Coefficients (DSCs) on a per-case basis. A Mean Absolute Error (MAE) of 0.03 is achieved on 1,610 test samples and binary classification accuracy of 97% is reported when separating low and high quality segmentations using a threshold on DSC.

Secondly, in the scenario where no manually annotated data is available, a network is trained to predict DSC from estimated quality obtained via the reverse testing strategy presented in chapter 3: $\mathrm{MAE} = 0.14$ and 91% binary classification accuracy are reported for this case. Predictions are obtained in real-time which, when combined with real-time segmentation methods, enables instant feedback on whether an acquired scan is analysable while the patient is still in the scanner. This further enables new applications of optimising image acquisition towards best possible analysis results.

The work in this chapter was published in the proceedings of the 21st annual conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), October 2018 [132].

4.1 Overview.

Finding out that an acquired medical image is not usable for the intended purpose is not only costly but can be critical if image-derived quantitative measures should have supported clinical decisions in diagnosis and treatment. Real-time assessment of the downstream analysis task, such as image segmentation, is highly desired. Ideally, such an assessment could be performed while the patient is still in the scanner, so that in the case an image is not analysable, a new scan could be obtained immediately and even automatically. Such a real-time assessment requires two components, a real-time analysis method and a real-time prediction of the quality of the analysis result. The work in this chapter proposes a solution to the latter with a particular focus on image segmentation as the analysis task.

This chapter follows the work laid out in chapter 3 where a reverse testing strategy was employed to assess the quality of biomedical image segmentations at large scale in the absence of Ground Truth (GT). The validation of a modified Reverse Classification Accuracy (RCA) approach and its application to automated Random Forest (RF) segmentations from the UK Biobank (UKBB) achieved promising results, but at the cost of an execution time that is only suitable for offline Quality Control (QC): $\sim$10 minutes per test segmentation.

The work detailed in this chapter will explore the idea of direct predictions of segmentation quality as well as predicting predictions made by RCA. It is validated on automated RF and Convolutional Neural Network (CNN)-generated segmentations of UKBB images which have manual GT labelmaps.

4.2 Introduction.

This work has so far considered the automated QC of biomedical image segmentations at large-scale, in the absence of GT. It has focused on methods which are inherently intuitive and that do not rely on Deep Learning (DL). However, it is clear that RCA can be a computationally intensive methodology owing to the number of image registrations and corresponding labelmap warpings. Additionally, it is seen that the time taken to evaluate a single segmentation with RCA is too long to be useful in automated clinical pipelines.

The work in this chapter aims to leverage the power of DL in order to assess the quality of automatically generated biomedical image segmentations. DL-based models may not always be straight-forward to train as they are sensitive to initialization parameters and dataset statistics, but at inference time they can operate extremely quickly, in the order of milliseconds. Though it may take longer to train a reliable DL-model and require the computational power of specialized hardware such as Graphics Processing Units (GPUs) to perform gradient-based update operations of the model, they can run efficiently on Central Processing Units (CPUs) at test time. In the clinical environment, this may be advantageous to allow quick, background processing of segmentations.

Recent advances in DL-based image segmentation have brought highly efficient and accurate methods, most of which are based on CNNs. However, even the best method will occasionally fail due to insufficient image quality due to noise, artefacts and corruption, or show unexpected behaviour on new data. In clinical settings, it is of paramount importance to be able to detect such segmentation failures on a per-case basis. In large-scale clinical research, such as population studies, it is important to be able to detect failure cases in automated pipelines, so invalid data can be discarded in the subsequent statistical analysis.

This study aims to assess the quality of automatically generated segmentations of Cardiac Magnetic Resonance Imaging (CMR) from the UKBB Imaging Study [123]. Refer to section 3.4.0.1 on page 96 for information on the UKBB.

4.2.1 Contributions.

This work shows that applying a modern DL approach to the problem of automated QC in deployed image-segmentation frameworks can decrease the per-case analysis time to the order of milliseconds whilst maintaining good accuracy. Dice Similarity Coefficient (DSC) is predicted at large-scale analyzing over 16,000 segmentations of images from the UKBB. Results show that measures derived from RCA can be used to inform the model removing the need for a large, manually-annotated dataset. When pairing this proposed approach for real-time quality assessment with real-time segmentation methods, new avenues of research can open up including automatic optimisation of image acquisition toward the best possible analysis results.

4.3 Related Works.

Automated QC is dominated by research in the natural-image domain and is often referred to as Image Quality Assessment (IQA). The literature proposes methodologies to quantify the technical characteristics of an image, such as the amount of blur, and more recently a way to assess the aesthetic quality of such images [159]. In the medical imaging domain, IQA is an important topic of research in the fields of image acquisition and reconstruction where anatomical coverage is also considered. Where research is conducted into the quality or accuracy of image segmentations, it is almost entirely assumed that there is a manually annotated GT labelmap available for comparison. This domain has seen little work on assessing the quality of automatically generated segmentations, particularly on a per-case basis, and in the absence of GT.

Manual IQA in large-scale studies is clearly infeasible. Initial work on automating IQA in CMR includes a Field-of-view (FOV) assessment algorithm [178]. Here, CNNs were used to identify missing apical and basal slices in a test set after training on the UKBB pilot dataset. Such errors directly lead to degraded performance of automated segmentation algorithms. More recent work [160] goes further, assessing heart coverage, inter-slice motion and image contrast in the cardiac region on almost 20,000 UKBB CMR.

Quality-Net [79] was proposed for assessing segmentation quality in the natural image domain. Three CNNs were presented: one taking only the segmented image, a second which segments the image and estimates the Intersection over Union (IoU), and a final network introducing the 'weighted mask' layer which extracts useful features from the background as well as foreground of the image. The work shows that the GT segmentation is not required at test time and the method can be performed on a per-case basis. Quality-Net has not been used in the medical imaging domain, possibly because of the complexity of medical imaging data. It is also common for multiple classes to be present in an image, especially in medical data, which Quality-Net is not explicitly designed to accommodate.

Other work in [183] proposes to train two CNNs for the task of segmentation quality assessment: the first to reconstruct the input image given a segmentation-masked copy and another to perform quality metric regression on difference images between the true and reconstructed input images. The method could be applied to medical images, though training DL-based models requires a large amount of annotated training data to avoid over-fitting.

In [151], a 'double network' is created which takes the image and segmentation-masked image. Three of these networks are combined taking inputs of different scales around the segmented area in order to extract global and local features relating to the image and labelmap. IoU based on GT is predicted with good Linear Correlation Coefficient (LCC). Multi-scale methods for medical image segmentation have been explored [162], but have not been previously applied directly for QC. A multi-scale QC approach would be interesting to explore particularly for segmentations of fine

structures, *e.g.* in neuroimaging.

Rather than trying to directly predict the traditional evaluation metrics for segmentation quality, the work in [75] extracts pixel-level uncertainty information from CNN-based segmentors as a measure of confidence in the final segmentation. They find good correlation between DSC and the mean uncertainty for some CNN architectures. On the same theme, [138] attempts to train a segmentor which outputs voxel- and structure-wise uncertainties that are used to calculate performance metrics, *e.g.* IoU. Uncertainty measures are a promising method for automated QC of segmentations, but the methods in these works are only applicable if the labelmaps are generated with CNN-based models. In cases where the segmentor cannot be interrogated, or has not been constructed to make uncertainty information easily accessible, an alternative method must be used. Ideally, a method for automated QC should be applicable to segmentations generated by any model such that they may be compared.

The work in [179] uses CNNs to learn a model for predicting the error in a given labelmap. It defines a Quality Indicator (QI) metric for overall segmentation quality by averaging the predicted per-pixel error. Their results show good correlation between QI and segmentation accuracy based on GT. This method shows promising results using training data generated by an off-the-shelf segmentor, achieving segmentations of varying quality by taking the side-outputs from the segmentor. The method performs segmentation QC in the absence of GT on a per-case basis and is able to work with segmentations generated by different models. One disadvantage here is that the learned error-map predictor learns only about labelmaps produced by the chosen automated segmentation model, rather than being more general. It cannot offer a comparison to similar segmentations or their source images.

Further use of CNNs for segmentation QC is found in [102] where a Variational Autoencoder (VAE) is trained on image-segmentation pairs to learn 'shape-priors' in a low-dimensional feature space. On evaluation, poor segmentations do not fit with this learned representation and can be identified. A regressor is also learned on the low dimensional feature space to predict DSC of a given labelmap.

This work aims to directly predict traditional segmentation quality metrics, *e.g.* DSC, in real-time and in the absence of GT. The methods are applied at large-scale using population imaging data from the UKBB. The aim is to leverage the inference speed of CNNs to give fast and accurate predictions of segmentation quality. This would be vital for analysis pipelines which would benefit from the increased confidence in segmentation quality without compromising processing time.

4.4 Methods and Materials.

Throughout this work the DSC is used as the metric of quality for segmentations. It is a tradiational quality metric for segmentations even in the medical imaging domain. As such, it is intuitive, commonly reported and well-understood. In comparison to some other DL-based QC methods, this will allow researchers and clinicians alike to gain an immediate intuition of the utility of the method. DSC measures the overlap between a proposed segmentation and its GT, which is usually a manual reference. It is described further in section 3.4.2 on page 99.

To investigate CNNs for automated QC, two experiments are conducted. The first aims to directly predict the DSC of segmentations in the absence of GT. The second considers the RCA method and aims to predict the outcome of the RCA process. The dataset and training scheme is laid out in this section followed by the experimental design.

4.4.1 Dataset.

The initial dataset consists of 4,881 3D (2D-stacks) Short-Axis (SAX) End-Diastolic (ED) CMR scans from the UKBB Imaging Study[1]. All images have a manual segmentation which is unprecedented at this scale [122]. The labelmaps available under this application act as GT. Each labelmap contains 3 classes: Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM) and Right Ventricular Cavity (RVC) which are separate from the background class (BG). In this work, the segmentation is also considered as a single binary entity comprising all classes referred to as Whole-Heart (WH).

A RF of $T = 350$ trees and maximum depth $D = 40$ is trained on 100 cardiac atlases from the Hammersmith dataset as previously described in section 3.4.3.1 on page 102. The trained RF is used to segment the 4,881 images at 13 depths: 2, 4, 6, 8, 10, 15, 20, 24, 28, 32, 36, 38 and 40. This generates segmentations at varying quality across the DSC range. Examples of segmentations generated at different depths are shown in figure 4.7 on page 133. After cleaning corrupted and zero-sized files caused by the segmentor, the DSC is calculated from the manual contours acting as GT for 61,574 generated segmentations. The distribution of DSC is shown in figure 4.1 on the following page with a bin size of 0.1 DSC. Due to the imbalance of DSC scores in this data, a random subset of segmentations is taken from each DSC bin, equal to the minimum number of counts-per-bin (1,610) across the distribution. The final dataset comprises 16,100 DSC-balanced segmentations with reference GT.

From each segmentation, 4 one-hot-encoded masks are created: masks 0 to 3 correspond to the classes BG, LVC, LVM and RVC respectively. The voxels of the i^{th} mask are set at $[0, 0, 0, 0]$ when they do not belong to the mask's class and the i^{th} element set to 1 otherwise. For example, the mask for LVC is $[0, 0, 0, 0]$ everywhere except for voxels of the LVC class which are given the value

[1] UK Biobank Resource under Application Number 2964.

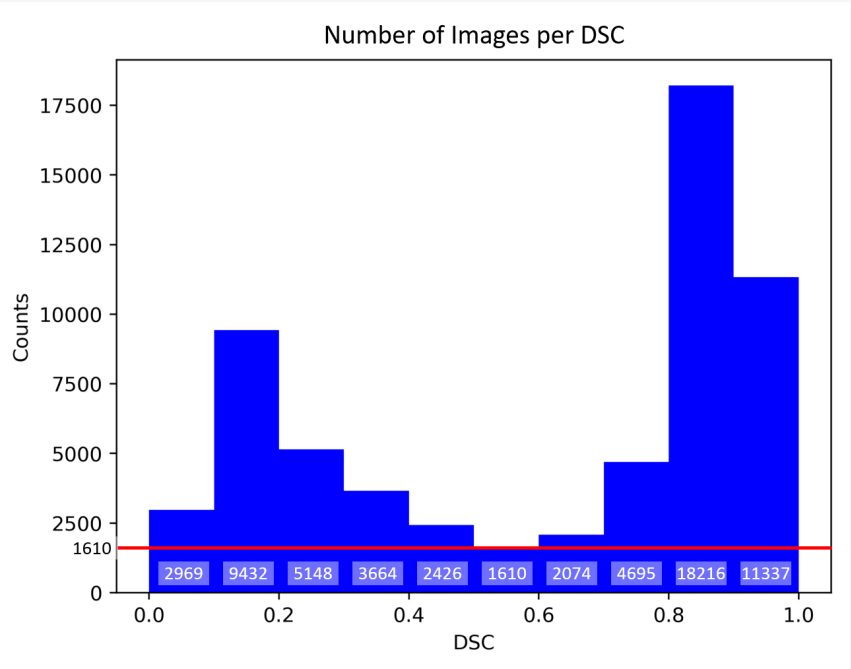


Figure 4.1: Histogram of DSC for 61,574 RF Segmentations in Experiment 1. The DSC range is $[0, 1]$ with 10 equally spaced bins of 0.1 DSC. Red line shows minimum counts (1,610) in the bin $\mathrm{DSC} = [0.5, 0.6)$ used to balance scores.

$[0, 1, 0, 0]$. Separating the segmentation masks will give the network a greater chance to learn the relationships between the voxels' classes and their locations. An example of the segmentation masks is shown in figure 4.2 on the next page.

At training time, the data-generator re-samples the images and segmentations to have consistent shape of $[224, 224, 8, 5]$ making the network fully 3D with 5 data channels: the image and 4 segmentation masks. The images are also normalized such that their intensities fall in the range $[0.0, 1.0]$.

4.4.2 Training.

For this work, a 50-layer 3D Residual Network (ResNet) [70] was written in Python using the Keras library. ResNets are advantageous as they allow the training of deeper networks by repeating smaller blocks. They benefit from skip connections that allow data to travel deeper into the network without losing information.

The ResNet takes the image-segmentation of shape $[224, 224, 8, 5]$. A series of operational blocks applies convolutions, Batch-normalization (BN) and Rectified Linear Unit (ReLU) activations to their input. Layers before and after each block are summed via the characteristic skip-connections before being passed into the next. The blocks perform max-pooling operations to reduce the size of the input until it reaches the final layers of the network. To finish, the network performs average pooling and constructs a Fully-connected (FC) layer which connects to the final

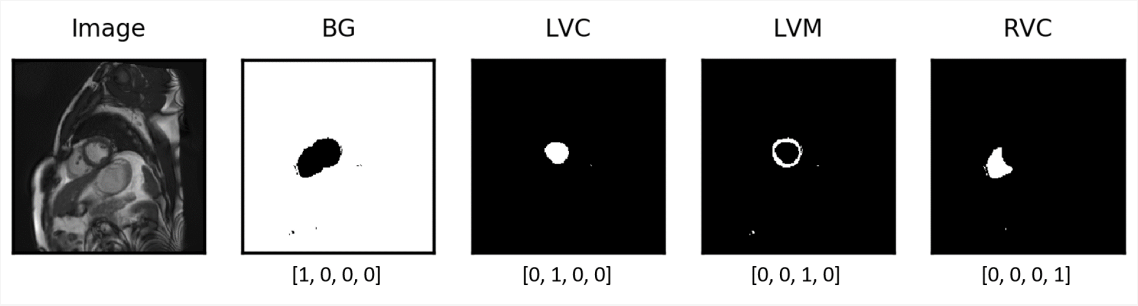


Figure 4.2: Sample CNNs Input for Experiments 1 and 2. The 5 channels of the input are the image, and one-hot-encoded labelmaps for the background (BG), Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM) and Right Ventricular Cavity (RVC) classes.

output layer. The output from the network has shape $[5]$, one node for each class in the segmentation mask.

For comparison and consistency, the dataset described in section 4.4.1 on page 123 and the network architecture described here are used for each experiment in this chapter.

The purpose of the model is to perform predictions of continuously valued DSC. Therefore the CNNs work as regressors. As such, no activation function is applied to the network output. For training, the Mean Squared Error (MSE) between the network predictions and the GT values is minimized. The Adam optimizer is employed during training with a learning rate of $1e^{-5}$ and learning rate decay of 0.005. Batch sizes are kept constant at 46 samples per batch. Validation is conducted at the end of each epoch for model evaluation and checkpointing. Experiments are conducted on 11 GB Nvidia GeForce GTX 1080-Ti GPUs.

4.4.2.1 Experiment 1: Predicting the DSC.

In the scenario where an automated segmentation algorithm is deployed in some large-scale, fast-paced analysis pipeline, it is important to perform QC quickly and accurately. To reduce the computation required, it would be ideal to use only the image-segmentation pair to calculate the quality metrics. It is also important to calculate metrics that are intuitive. This experiment aims to predict the DSC using only the image-segmentation pair. The DSC is a well-known metric and is ubiquitous in quality assessment.

In this experiment, the DSC is first calculated per class between the 16,100 generated segmentations and their corresponding manual contours. These are used as labels during training.

The final layer of the network has output $X \in \mathbb{R}^5$ with $x_i \in [0.0, 1.0] \, \forall \, x_i \in X$. This vector represents the DSC per class including BG and WH. MSE loss between the predicted and calculated DSCs per class is minimized during training.

The data is split 80:10:10 giving 12,880 training samples and 1,610 samples each for validation and

testing. The Mean Absolute Error (MAE) is reported between the output and GT DSC.

Performing this experiment is costly as it requires a large manually-labeled dataset for training which is not readily available in practice. However, it is an important experiment to conduct here to ensure that it is possible to predict the DSC as a proxy for quality before implementing the second approach in this work.

4.4.2.2 Experiment 2: Predicting QC Scores from RCA.

Experiment 1 requires a large, manually labelled dataset in order to calculate the GT DSC for training. Additionally, predictions of segmentation quality may benefit from comparisons with other segmentations, rather than predicting a single score directly. The Reverse Classification Accuracy (RCA) framework validated in chapter 3 has shown promising results in accurately predicting the quality of segmentations of large datasets in the absence of GT. It uses a reference set of manually segmented images to inform the prediction. Experiment 2, here, aims to leverage this extra information by using the predictions from RCA as training data to allow a CNN to give comparatively accurate predictions on a test-set.

In this experiment, RCA is performed on all 16,100 segmentations. To ensure that the network is trained on balanced scores, histogram binning is done on the RCA scores as described for Experiment 1 in section 4.4.1 on page 123. An equal number of samples is taken from each class. This results in a total of 5,363 samples split into training, validation and test sets of 4,787, 228 and 228 respectively. The predictions per-class are used as labels during training. Similar to Experiment 1 in section 4.4.2.1 on the preceding page, a single predicted DSC output is obtained for each class using the same network and hyper-parameters.

The primary advantage of this approach is that there is no need for the large, often-unobtainable manually-labeled training set. Any user can perform RCA on a dataset using only the small set of reference images, and use a network trained on the RCA scores to gather predictions on new data in the same domain. This can be envisioned within a clinical pipeline or pharmaceutical trail where the incoming data stream is similar to data that have previously been obtained and processed.

4.5 Results.

4.5.1 Experiment 1 Results: Predicting the DSC.

Table 4.1: Results showing MAE from Experiment 1: directly predicting DSC MAE for poor (DSC < 0.5) and good (DSC ≥ 0.5) quality segmentations over individual classes and WH. Standard deviations in brackets. Statistics are shown for binary classification with threshold DSC $= 0.7$: True Positive Rate (TPR) and False Positive Rate (FPR) over full DSC range with classification accuracy (Acc).

	Mean Absolute Error (MAE)		
	$0 \leq$ DSC ≤ 1	DSC < 0.5	DSC ≥ 0.5
Class	$n = 1,610$	$n = 817$	$n = 793$
BG	0.008 (0.011)	0.012 (0.014)	0.004 (0.002)
LV	0.038 (0.040)	0.025 (0.024)	0.053 (0.047)
LVM	0.055 (0.064)	0.027 (0.027)	0.083 (0.078)
RVC	0.039 (0.041)	0.021 (0.020)	0.058 (0.047)
WH	**0.031 (0.035)**	**0.018 (0.018)**	**0.043 (0.043)**

TPR $= 0.975$ FPR $= 0.060$ Acc. $= 0.965$

Results from Experiment 1 are shown in table 4.1. The MAE between predicted DSC and DSC calculated against reference GT is reported per class along with standard deviations. Results show that the network can directly predict WH DSC from the image-segmentation pair with MAE $= 0.03$ ($\sigma = 0.04$). Similar performance is observed on individual classes. The LVM has the highest MAE (0.055) which corresponds to findings in chapter 3 on page 88. The same reasoning can be applied here: the LVM has a greater variation when segmented as it has 2 boundaries and irregular shape, particularly at the base of the heart. Therefore, it is difficult to predict what the DSC will be as it depends on the accuracy of the manual GT contours. However, the MAE and standard deviation for the LVM are still small.

Overall, the method shows excellent accuracy of 96.5% when a binary threshold is applied to the WH case with 'poor' segmentations having DSC < 0.7 and 'good' segmentations DSC ≥ 0.7. The reported accuracy (96.5%) is better than the 95% reported with RCA in section 3.5 on page 106. Results show excellent True Positive Rate (TPR) and False Positive Rate (FPR) on the WH binary classification task.

Table 4.1 also shows MAE over the top and bottom halves of the GT DSC range. This suggests that the MAE is equally distributed over poor and good quality segmentations. The WH case is visualized in figure 4.3 on the next page (right) showing the MAE for both halves follow similar distributions to the full range. This demonstrates the ability of the network to predict segmentation

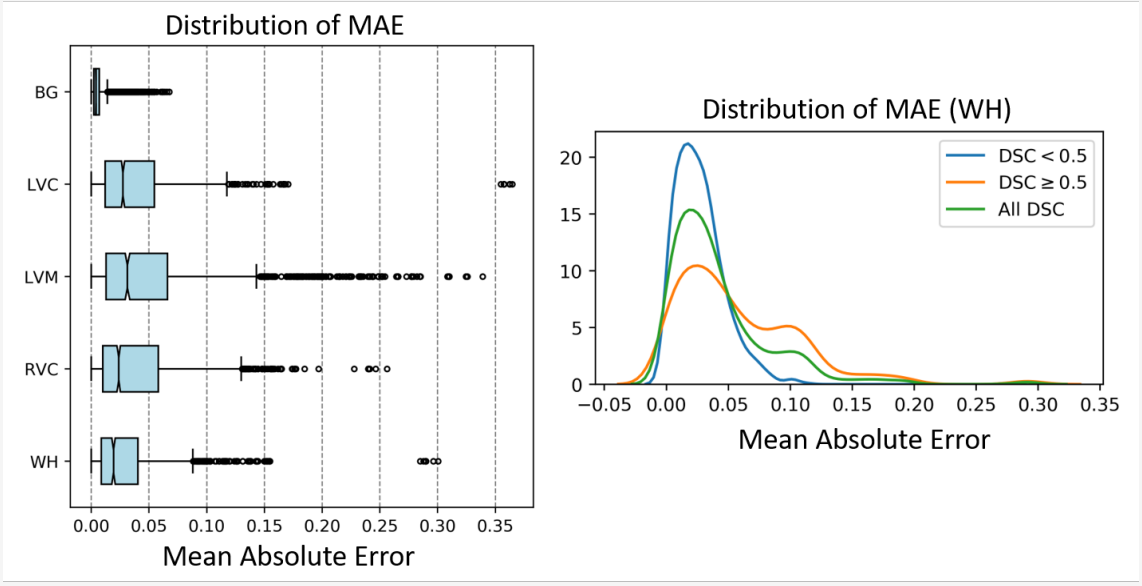


Figure 4.3: Quantitative Results showing MAE from Experiment 1. (left) The Mean Absolute Error (MAE) distribution over all segmentations in Experiment 1 separated by class (top-to-bottom): Background (BG), Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), Right Ventricular Cavity (RVC) and Whole-Heart (WH). (right) The MAE distribution for the WH case over the lower DSC < 0.5 and upper DSC ≥ 0.5 halves of the Dice Similarity Coefficient (DSC) range.

accuracy (DSC) when the segmentation is both very good, and very poor. A network that was only able to predict good quality segmentations would only be useful for identifying those that had failed by some threshold. However, in this case, it is possible to examine the extent to which a segmentation has been unsuccessful along the DSC continuum.

For the WH case 72% of the data have MAE < 0.05 with outliers (defined as MAE ≥ 0.12) comprising only 6% of the data. Distributions of the MAE for each class can be seen in figure 4.3.

For visual reference, examples of good and poor quality segmentation predictions are shown in figure 4.4 on the next page with their GT and predictions.

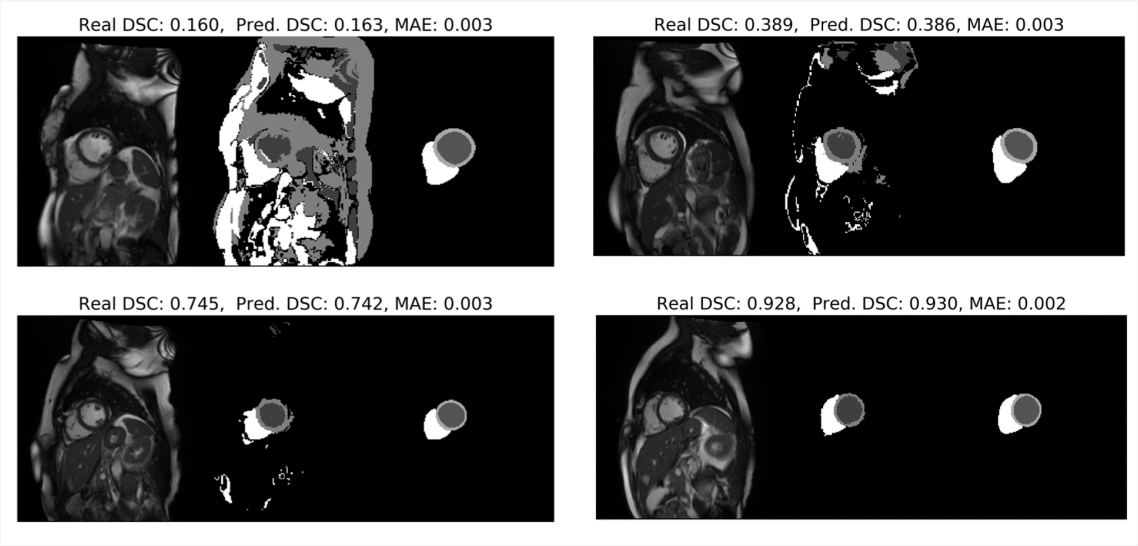


Figure 4.4: Qualitative Results from Experiment 1. Examples showing excellent prediction of DSC in Experiment 1. Quality increases from top-left to bottom-right. Each panel shows (left to right) the image, test-segmentation and reference GT.

4.5.2 Experiment 2 Results: Predicting scores from RCA.

The results from RCA were predicted on the test set and results recorded in table 4.2 on the following page. Results show MAE $= 0.14$ ($\sigma = 0.09$) when predicting the RCA result on the WH case with similar results on the other classes. Again, the LVM shows the highest MAE and standard deviation as in Experiment 1 and in section 3.5 on page 106 of the previous chapter. Overall, the standard deviations for each class are higher than when predicting the DSC directly, showing a greater uncertainty in the predictions.

On the binary classification task with a threshold at DSC $= 0.7$, an accuracy of 91% is achieved. The TPR $= 0.879$ is good and the FPR is negligible. In this experiment, there is a greater MAE on the DSCs in the upper half of the full range compared with Experiment 1. This indicates that the predictions are generally more reliable for the poor quality segmentations. In the deployed setting, the binary threshold applied to the WH would achieve good performance for automated QC. Rather than be used to predict good quality on individual classes, this method would be used to identify particularly poor quality WH cases. Distributions of the MAE are shown in figure 4.5 on page 131 for individual classes and the WH case. The distribution is heavily centered in the low MAE region with few of outliers.

Examples of a good and poor quality segmentation as predicted by RCA are shown in figure 4.6 on page 131 along with the CNN predictions.

It is expected that direct predictions of DSC from the RCA labels are less accurate than in Experiment 1. The reasoning is two-fold: first, the RCA labels are themselves *predictions* and retain inherent uncertainty. Second, the training set here is much smaller than in Experiment 1.

Table 4.2: Results showing MAE from Experiment 2: predicting RCA scores. MAE for poor (DSC < 0.5) and good (DSC ≥ 0.5) quality segmentations over individual classes and WH. Standard deviations in brackets. Statistics are shown for binary classification with threshold DSC $= 0.7$: True Positive Rate (TPR) and False Positive Rate (FPR) over full DSC range with classification accuracy (Acc).

	Mean Absolute Error (MAE)		
	$0 \leq \text{DSC} \leq 1$	$\text{DSC} < 0.5$	$\text{DSC} \geq 0.5$
Class	$n = 1,610$	$n = 817$	$n = 793$
BG	0.034 (0.042)	0.048 (0.046)	0.074 (0.002)
LV	0.120 (0.128)	0.069 (0.125)	0.213 (0.065)
LVM	0.191 (0.218)	0.042 (0.041)	0.473 (0.111)
RVC	0.127 (0.126)	0.076 (0.109)	0.223 (0.098)
WH	**0.139 (0.091)**	**0.112 (0.093)**	**0.188 (0.060)**

$\text{TPR} = 0.879 \quad \text{FPR} = 0.000 \quad \text{Acc.} = 0.906$

This approach would be a valuable addition to an analysis pipeline where operators can be informed of likely poor-quality segmentations, along with some confidence interval, in real-time. The benefit here is that there is no need for a large, manually-labelled training set: the labels can be generated by running RCA on the dataset.

On average, the inference time for each network was of the order 600 ms on CPU and 40 ms on GPU. This is over 10,000 (GPU) and 1,000 (CPU) times faster than with RCA (660 seconds) whilst maintaining good accuracy. In an automated image analysis pipeline, this method would deliver excellent performance at high-speed and at large-scale. When paired with a real-time segmentation method it would be possible provide real-time feedback during image acquisition whether an acquired image is of sufficient quality for the downstream segmentation task.

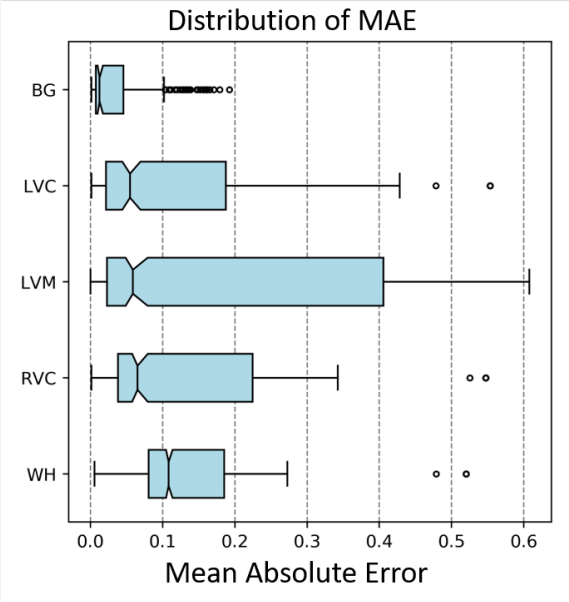


Figure 4.5: Quantitative Results showing MAE from Experiment 2. The Mean Absolute Error (MAE) distribution over all segmentations in Experiment 2 separated by class (top-to-bottom): Background (BG), Left Ventricular Cavity (LVC), Left Ventricular Myocardium (LVM), Right Ventricular Cavity (RVC) and Whole-Heart (WH).

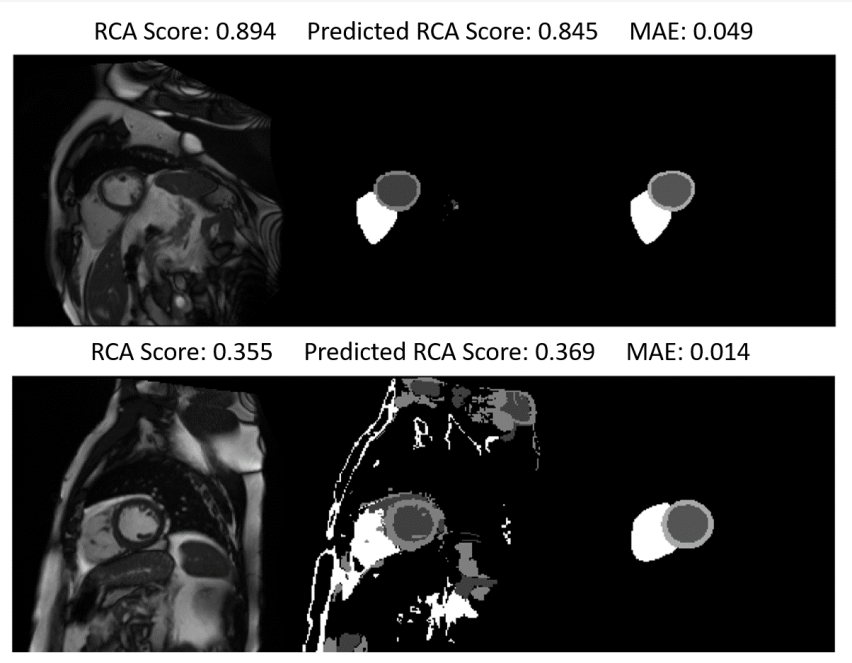


Figure 4.6: Samples of Segmentation QC by RCA and the QC CNN in Experiment 2. The panels show (left-to-right) the CMR scan, automatic RF segmentation, and GT labelmap for a (top) good and (bottom) poor quality segmentation as predicted by RCA. The CNN in Experiment 2 predicts the RCA score well, but does better in the poor quality case.

4.6 Conclusions.

Ensuring the quality of an automatically generated segmentation in a deployed image analysis pipeline in real-time is challenging. These experiments have shown that CNNs can be employed to tackle the problem with great computational efficiency and with good accuracy.

A The CNNs may learn features specific to the RF segmentor.

It is possible that the networks in both experiments are prone to learning features specific to assessing the quality of RF segmentations. The segmentations generated by RF classifiers are distinctive: with decreasing depth, the forest generates more false-positives in the labelmap which extend further from the relevant anatomical features. Examples of this can be seen in figure 4.7 on the following page. A simple way to assess the quality of these CMR segmentations is to find the number of non-BG labelled voxels which are furthest from the central mass of the mask, *i.e.* the heart. Of course, a RF trained for multi-organ segmentation or segmentation of disparate entities cannot be assessed in this way. This would be an easy task for a CNN to learn. As such, this may account for the decreased error and uncertainty in the RCA predictions of low quality segmentations in Experiment 2.

To generalize these QC CNNs, they could be trained with segmentations generated from an ensemble of methods: RF, multi-atlas, CNN *etc.* However, it must be reiterated that the purpose of the framework in this study is to give an indication of the *predicted quality* and not a direct one-to-one mapping to the reference DSC. Currently, these networks will correctly predict whether a segmentation is 'good' or 'poor' on some threshold, but will not confidently distinguish between two segmentations of similar quality.

B The CNNs are currently not suitable for assessing fine-grain segmentation details.

The trained CNNs are insensitive to small regional or boundary differences in labelmaps which are of good quality. Thus they cannot be used to assess the quality of a segmentation at fine-scale. Segmentations of volumes, such as the WH, appear to be less susceptible to the prediction errors. The class with the greatest prediction error is the LVM. The LVM consists of two surface boundaries of irregular shape that enclose a small volume. It is clear that segmentations of this region will be more prone to the errors described which are also discussed in chapter 3 on page 88.

The problem may be improved by training the networks on a more diverse training-set with a focus on variations in LVM topology, particularly in difficult to segment regions such as the base of the heart. Providing small differences in the boundary of individual regions, rather than wholesale variations across the entire image, would give the networks a better understanding of what it is they are being asked to do. For example, it may be possible to train a network specifically for LVM QC by segmenting the LVM for each sample many times with slightly variations to the boundary in each one. The trained network could then provide finer predictions of these irregular structures.

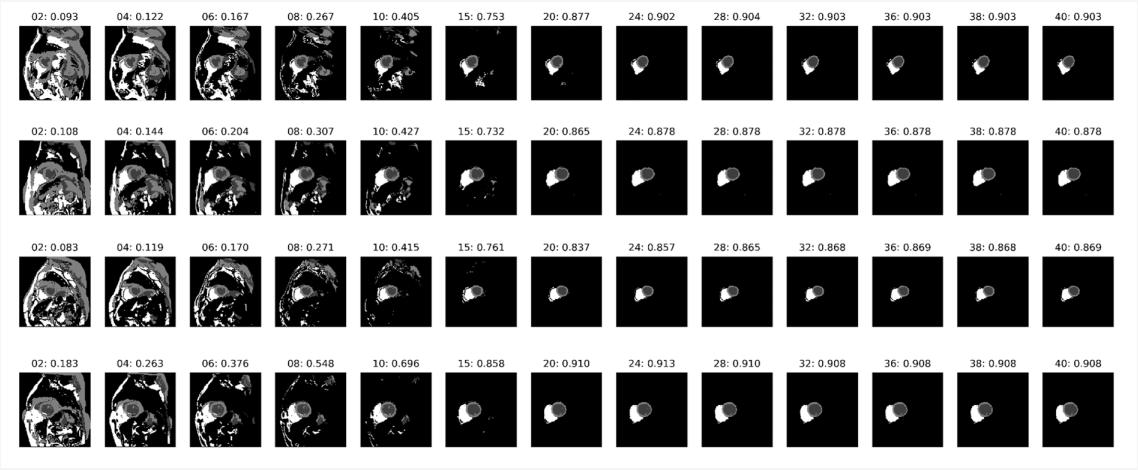


Figure 4.7: Samples of Random Forest (RF) Segmentation Degradation at Different Tree-depths. Each row is different sample image with segmentations generated using a trained RF. Tree depth and DSC indicated above each segmentation.

C A CNN can assess segmentation quality without large labelled training sets.

The contours on the training images for the network in Experiment 1 were drawn by a team of manual raters. Such labelled datasets are not easily available in most cases. However, RCA requires only a small reference dataset which has GT annotations. By performing RCA on a larger dataset, one can automatically obtain training labels for the network in Experiment 2. Modifying an automated segmentor to generate segmentations of varying quality, as is done in this work, would be beneficial. This could also be applied to segmentations generated with other algorithms, *e.g.* by limiting the depth or number of filters in a CNN segmentor.

The cost of using data obtained with RCA is an increase in MAE. The rise is not substantial for poor quality segmentations. This could be seen as a reasonable trade-off compared to the effort, time and monetary cost required to obtain a large, manually-labeled dataset.

D Methods cannot tell us about *why* a segmentation has failed.

A automated segmentation method may fail for many reasons. It was discussed in this work that image quality can have a big impact on segmentation quality: see section 3.3 on page 92 and section 4.3 on page 121. Neither RCA nor the CNNs trained in this work are able to provide reasons for the segmentation failure as they are currently implemented, they indicate only which segmentations may be of poor quality.

Some approaches to DL-based prediction model interrogation have been studied including feature visualization, attribution, and generative methods [145]. It would be useful to delve further into this area in order to identify salient regions of the images that have the greatest influence over segmentation quality.

A major contributor to segmentation failure is Domain Shift (DS) which arises when algorithms are trained on a dataset acquired from one scanner and evaluated on a dataset acquired from a different scanner or of a different population. The work in chapter 5 explores a method to mitigate the effects of DS on Machine Learning (ML) models.

Chapter 5

Image-level Harmonization of Multi-Site Medical Imaging Data using Image and Spatial Transformer Networks

Assessing the quality of a Machine Learning (ML) model's outputs is not enough to ensure its reliability: they must be robust enough to subtle variations in input data. ML models trained and tested on images acquired from different sites often exhibit a drop in performance. This Domain Shift (DS) problem is important when considering the use of multi-site medical imaging data. This chapter investigates the use of Image and Spatial Transformer Networks (ISTNs) to tackle DS in multi-site medical imaging data.

Commonly, Domain Adaptation (DA) is performed with little regard for explainability of the inter-domain transformation and is often conducted at the feature-level in the latent space. ISTNs are employed for unsupervised and unpaired DA at the image-level which constrains transformations to explainable appearance and shape changes.

As proof-of-concept, experiments are performed to demonstrate that ISTNs can be trained adversarially on a classification problem with simulated 2D data. For real-data validation, two 3D brain Magnetic Resonance Imaging (MRI) datasets are created from the Cam-CAN and UKBB imaging studies to investigate DS due to acquisition and population differences. Results show that age regression and sex classification models that are trained on the ISTN output improve generalization when training and testing on data from different sites. Individual transformations can be visualized to aid explainability.

The work in this chapter is accepted to the 23rd annual conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), October 2020 [131].

5.1 Overview

Machine Learning (ML) models are usually trained in an offline, idealised environment using large, high-quality and well-curated datasets. Models trained in this source domain are often deployed into environments where they are used to predict some statistics on smaller, lower-quality datasets which are bespoke to the target environment. It is expected that there will be a degradation in the performance of these models when they are moved between the source and target domains. However, when the source and target domains are very similar, or even related, this reduction in model performance can still occur. This Domain Shift (DS) problem in medical image analysis is the focus of this chapter.

In the medical imaging domain, training datasets are often the best sets of images available to researchers. They are commonly acquired from a single site and annotated by expert raters. Those datasets that are composed of images from multiple sites have the potential to introduce site-specific biases into models [166]. This work describes two forms of DS common to medical imaging datasets: acquisition shift and population shift.

The DS problem between multi-site imaging data must be addressed if powerful models are to be built and deployed for large scale population analysis. In this work, experiments are performed using data from neuroimaging datasets including the large scale UK Biobank (UKBB) Imaging Study [154].

DS can be tackled by performing Domain Adaptation (DA) as a form of Data Harmonization (DH). DA is a growing research area in medical image analysis and can be carried out in a number of ways. Research in the area of DA is extensive and terminology can vary wildly from paper to paper. Section 5.2 on the following page sets out the terminology used in the field which directly relates to research conducted in this work. An overview of the research conducted into DA is presented in section 5.3 by focusing on works in the medical imaging domain. The current state-of-the-art is described along with established methodologies. Methods and datasets used within this work are detailed in section 5.4 alongside the experimental setup. Finally, an overview and discussion of the results from these experiments is given in section 5.5 before conclusions on the work are presented in section 5.6.

5.2 Introduction.

Here, a background on Domain Shift (DS) is presented and its effects in the medical imaging domain are described, beginning with a review of common terminology and their definitions as they are applied in this work. Section 5.3 on page 142 reviews the literature related to Domain Adaptation (DA) which is the focus of this work.

5.2.1 Terminology and Definitions for Domain Adaptation.

The field of DA is peppered with interlinking and subtly different terminology. To avoid confusion, a set of terms and definitions related directly to this work is presented here.

1 Domain Shift.

In most literature, a 'domain' $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ with $X = \{x_1, x_2, ..., x_i\} \subset \mathcal{X}$ [119]. Domains may differ in feature space or marginal probability distributions. For example, a set of images x_i^1 drawn from $P(X_1)$ is different to the set x_i^2 drawn from $P(X_2)$ but they share the same feature space $\mathcal{X}$. Thus, two datasets are considered to be from different domains if their marginal distributions differ.

Randomly splitting a single dataset into subsets, as is done during cross-validation, should not change the statistics or distribution of the data within each subset - they are considered to be from the same domain. However, a new dataset collected for evaluation may exhibit changes in appearance, *e.g.* image subjects; population statistics. *e.g.* age of a population sample; time periods, *e.g.* night and day; or other non-linear global changes, *e.g.* device-specific biases. This new dataset is considered to be a different domain.

DS is the problem that arises when models trained on a source domain $\mathbf{S}$ are tested on data from a different target domain $\mathbf{T}$ with different characteristics. DS is often accompanied by a drop in model performance.

DS is also described in relation to the task being performed. A source model $\mathcal{T}$ may be trained to perform some task, but may be used to perform a different, but related, task. For example, a model trained to identify lemons may be a useful starting point in trying to identify limes. In this case, the DS is between task domains rather than between datasets.

2 Dataset Shift.

DS that specificly realtes to changes between datasets is referred to as dataset shift. This term was used first in [128] which collects a series of papers on the topic. It states that dataset shift occurs when the joint distribution of inputs and outputs differs between training and test stages; see equation (5.1) on the following page. The most common form is 'covariate shift' where only the input distribution changes. Other terms for the same idea include 'concept

drift' [172], 'changes of classification' [169], 'changing environments' [3] and 'fracture points' [28]. This work concerns DS specifically in relation to dataset shift.

$$P_{\text{training}}(\mathbf{y}, \mathbf{x}) \neq P_{\text{testing}}(\mathbf{y}, \mathbf{x}) \tag{5.1}$$

3 Transfer Learning.

Transfer Learning (TL) aims to improve the learning of a predictive function $\mathcal{T}_T$ in the target domain $\mathcal{D}_T$ using knowledge from the source domain $\mathcal{D}_S$ and source task $\mathcal{T}_S$. The source and target domains and tasks are all different. A widely used example of TL is the use of all but the classification layer of an ImageNet-trained [42] classifier $\mathcal{T}_S$ to speed-up the learning of classes in a new domain $\mathcal{D}_T$. The features learned on the ImageNet dataset $\mathcal{D}_S$ help the new classifier $\mathcal{T}_T$ to learn more quickly.

4 Domain Adaptation.

DA is a transductive TL technique [119] that aims to modify the source domain's marginal distribution of the feature space such that it resembles that of the target domain. Therefore, when two domains are different, but related, DA can be applied to make them more similar. It is usually the case that there are labels associated with the data in the source domain, but labels are absent or scarce in the target domain. This scenario describes 'unsupervised' DA. It is unlikely that a sample in the source domain will have a corresponding sample in the target domain. For example, in the case where we want to transfer from 'daytime' to 'night-time' image domains, we cannot assume that for every photograph taken at a specific location in the daytime dataset, there exists a corresponding photograph taken at night-time in the night-time dataset. This common scenario is referred to as 'unpaired' DA.

5 Domain Mapping.

DA models that directly transform samples from the source to target domain learn a 'mapping' from one to the other. In Domain Mapping (DM) approaches, the models are often Image-to-Image (I2I) translation networks where the input is an image from the source domain and the output is the same image but transformed to align more closely with the statistics of the images in the target domain. These approaches use an adversarial training method which works to force the output from the DA model to be similar to those in the target domain [174]. This work employs an adversarial unsupervised domain mapping approach.

For brevity: Domain Mapping is a type of Domain Adaptation approach where the domains are different, but related. Unsupervised Domain Adaptation is a subset of Transfer Learning approaches where labels are available only in the target domain. All of these methods try to address the Domain Shift problem, but this work focuses specifically on dataset shift.

5.2.2 Domain Shift in Medical Image Analysis.

DS can cause a significant drop in the predictive performance of a Machine Learning (ML) model. It has been observed in almost all recent medical imaging challenges where the final test data came from different clinical sites [38]. DS is a major hurdle for successfully translating predictive models into clinical routine. Acquisition and population shifts are two common forms of DS that appear in medical image analysis [23].

Acquisition shift.

Acquisition shift is observed due to differences in imaging protocols, modalities or scanners. It may be clear that prediction models trained on Computed Tomography (CT) images will not give the same level of performance on Magnetic Resonance (MR) images; the features at the image level are very different. Acquisition shift will also be observed even if the same subjects are repeatedly scanned in the same scanner, but with different acquisition parameters, *e.g.* Magnetic Resonance Imaging (MRI) pulse sequences. This form of DS is particularly important to consider for longitudinal studies of a subject or cohort. Naïvely, predictive models trained on images acquired with the old acquisition parameters may be evaluated on the new images. It is likely that DS between the two sets of images will cause degradation in the model performance.

Population Shift.

Population shift occurs when cohorts of subjects under investigation exhibit different statistics, resulting from variations in demographics or disease prevalence. Medical imaging datasets are often very small in comparison to their natural image counterparts and it is not uncommon for each dataset to come from a single site [94]. It is also likely that the medical imaging datasets were created to study cohorts of subjects with specific characteristics, such as those with particular diseases or of a particular age. These factors contribute to the population shift that occurs when predictive models are trained on one of dataset and evaluated on another.

It is not uncommon for both acquisition and population shift to occur simultaneously, especially in multi-center studies. It is essential to tackle DS in ML to perform reliable analysis of large populations across sites and to avoid introducing biases into results.

Recent work has shown that even after careful pre-processing of medical imaging datasets, site-specific differences remain in the images [56, 166]. Both works observed that classifiers can easily be trained to predict the scanner from which an image was acquired. Experiments in [56] showed that certain pre-processing steps such as image whitening and spatial normalization can increase the confidence of the site-classifier. It was observed that the classifier can become in-

creasingly confident with greater levels of spatial normalization in the imaging data. Experiments were also performed with sex-imbalanced datasets. Sex-classifiers trained on improperly site-balanced datasets fall back to being site-classifiers when the female to male ratio differs between domains. This causes a substantial performance drop of the task model when trained on one domain and tested on the other.

It is clear that confounding biases in medical imaging datasets are an obstacle that must be overcome in order to ensure that ML models are accurate and reliable. In deployment of these models, it is vital that clinicians and other stakeholders are assured of a model's ability to perform well in their individual settings.

An approach to harmonizing the training and test sets is to apply DA such that the images from one site become more like images from the other. This can be achieved by applying transformations to the images in order to balance out variations in intensity and spatial deformations. Image-level transformations have the advantage of inherent visual explainability. In this work, DA at the image level is considered.

5.3 Related Works

This section outlines work which discusses Domain Adaptation (DA) and its application to the medical imaging domain. Recent works are described and their relation to this work is assessed. The methodology adopted in this work is framed in the current research landscape before being described in detail in section 5.4 on page 145.

5.3.1 Domain Adaptation in Medical Image Analysis.

In medical image analysis, datasets acquired from two different sites or scanners may be considered to be from different domains. The images may follow different distributions due to intrinsic differences between the scanners used to acquire the images, *i.e.* acquisition shift, or the population of subjects being imaged, *i.e.* population shift. These are described in section 5.2.2 on page 140. Using DA to transform data between the domains may help to overcome these site-specific biases and recover model performance that is lost due to Domain Shift (DS).

In medical imaging, labelled data are scarce and typically unavailable for the target domain. It is also unlikely to have the same subjects in both source and target domains. Thus, we focus on 'unsupervised' and 'unpaired' DA, wherein labelled data is available only in the source domain and no matching samples exist between source and target.

Many DA approaches focus on learning domain-invariant feature representations for the downstream task models. This is done either by forcing latent representations of the inputs to follow similar distributions, or 'disentangling' domain-specific features from generic features [177]. Learning domain-invariant features can be achieved with some divergence measure based on dataset statistics or by training adversarial networks to model the divergence between the feature representations [176]. These methods have been applied to brain lesions [89] and tumours [40] in MRI, as well as in contrast to non-contrast CT segmentation [142].

While these approaches seem appealing and have shown some success, they lack a notion of explainability as it is difficult to know what transformations are applied to the feature space. Additionally, although the learned task model may perform equally well on both domains, it is not guaranteed to perform as well as separate models trained on the individual domains.

This work explores model-agnostic DA by working at the image level. The approach is based on Domain Mapping (DM), which aims to learn the pixel-level transformations between two image domains, and includes techniques such as style transfer. Model-agnostic DA is the 'decoupling' of the DA process from any task-specific architectures [12]. Pix2pix [81] (supervised) and CycleGAN [184] (unsupervised) take images from one domain through some encoder-decoder architecture to produce images in the new domain. Details on CycleGAN as relates to this work can be found in section 2.2.4 on page 49. The method in [176] works on both the image and latent feature levels

and uses CycleGAN to improve segmentation across scanners.

In [12], the generator is conditioned on both a noise vector and the source domain images. The work also introduces a content-similarity loss which works on foreground pixels. The Residual Network (ResNet)-based model is trained with a traditional adversarial loss. The authors state that only low-level changes are expected between domains in their work, rather than higher level changes such as variations in geometry. The work presented in this chapter proposes the addition of a constrained spatial transformer to account for small geometry variations between images.

The literature around DA includes many works concerning the use of synthetic data [12, 129]. Synthetic data is often used for data augmentation purposes in the hopes of improving the generalization of trained models, however there is a discrepancy between the synthetic and real data. DA can be used to reduce the gap between the generated, synthetic data and the real distribution. This work focuses on DA between real datasets in a high-stakes environment where the DS may be caused by small, almost imperceptible differences between the images.

The work in [112] claims to have developed a 'general framework' for DA of which other methods are 'special cases'. However, it focuses on combining the developments in domain-invariant feature learning and pixel-level DA. This work is focused on the latter with constrained transformations in order to maximize explainability which is not straight-forward with domain-invariant feature learning.

Similarly, CyCADA [76] combines domain-invariant features with image-level DA by combining pixel- and feature-losses with semantic and cycle consistency losses. Whilst this harnesses the power of domain-invariant feature learning, some explainability of the image-level adaptation is lost. FlexAdapt [107] extends the CycCADA approach showing improvements where the label spaces in source and target domains only partially overlap.

Methods for DM primarily use UNet-like [137] architectures to learn Image-to-Image (I2I) transformations that are easier to interpret, as one can visually inspect the output. For medical images of the same anatomy, but from different scanners, we assume that DS manifests primarily in appearance changes (contrast, signal-to-noise ratio, and resolution), anatomical variation, and further subtle variations caused by the image reconstruction and interpolation processes. Therefore, this work proposes separating and constraining the spatial and appearance transformations using Image and Spatial Transformer Networks (ISTNs).

5.3.2 Contributions.

This work proposes the use of ISTNs [99] to tackle DS at the image-feature level in multi-site imaging data. ISTNs separate and compose the transformations for adapting appearance and shape differences between domains. This work is the first to use such an approach with retraining of the downstream task model on images transferred from source to target. This work shows that

ISTNs can be trained adversarially in a task model-agnostic way. The transferred images and their transformations can be visually inspected, and thus, the approach adds explainability to DA. This is important for validating the plausibility of the learned transformations. Results demonstrate the successful recovery of performance on classification and regression tasks when using ISTNs to tackle DS. Both unidirectional and bidirectional training schemes are assessed. This work compares retraining the task model from scratch versus finetuning. Proof-of-concept results on synthetic images generated with Morpho-MNIST [24] are presented for a 3-class classification task. The method is then validated on real multi-site data with 3D T1-weighted brain Magnetic Resonance Imaging (MRI). Results indicate that ISTNs improve generalization and that predictive performance can be recovered close to single-site accuracy.

5.4 Methods and Materials

This section first outlines the Deep Learning (DL)-based models used in this work to perform DA between multi-site medical imaging datasets. The specific parameters for the models employed are described and the training schemes are laid out.

5.4.1 Experimental Setup.

A predictor that is trained on one domain and evaluated on another will exhibit a degradation in model performance due to DS. Concretely, if $\mathcal{T}_S$ is a task model, such as a classifier or regressor, and is trained on the source domain $\mathbf{S}$, the performance of $\mathcal{T}_S(S)$ is likely to be the 'best performance' whilst the performance of $\mathcal{T}_S(T)$ on the target domain $\mathbf{T}$ will degrade due to DS.

5.4.1.1 Unsupervised Approach.

Most works that address the DS problem employ domain-invaraint feature learning DA methods. The work presented here makes use of ISTNs to learn explicit appearance and spatial transformations at the image level in order to perform DA. This is achieved by training the ISTN and simultaneously re-training the task model $\mathcal{T}_S$ on the ISTN output $S2T$. The resulting model $\mathcal{T}_{S2T}$ is trained to achieve maximum performance on the transformed images $\mathcal{T}_{S2T}(S2T)$ using the labels from S. To assess the performance 'recovery' of $\mathcal{T}_{S2T}$, $\mathcal{T}_S(T)$ and $\mathcal{T}_{S2T}(T)$ are compared. In practice, data from T would be unlabelled.

Having labelled data in the source domain $\mathbf{S}$ and unlabelled data in the target domain $\mathbf{T}$ defines this experimental setup as an 'unsupervised' DA problem [119]. For evaluation of the methods described in this work, datasets are used with labels in both domains.

Having 'paired' data is uncommon in medical imaging datasets. Such a dataset would require images of the same subject acquired in all domains under investigation, *i.e.* the same subject but different scanners or modalities. In this work, we use datasets with no repeat subjects in either domain, thus an 'unpaired' approach to DA is described.

5.4.1.2 Task Agnostic Training.

To avoid the introduction of artifacts and biases into the target domain during training of the task model, it is appropriate to transfer images S from the source domain $\mathbf{S}$ to the target domain $\mathbf{T}$ rather than transforming images T from $\mathbf{T}$ to $\mathbf{S}$. This approach ensures that the test data T from the new domain is not modified in any way.

Additionally, in scenarios where the original model $\mathcal{T}_S$ is deployed, it is likely to have been trained on a large, well-curated and high-quality dataset; it cannot be assumed that the same quality would be available for every new test domain.

One approach to training the ISTN is to update its parameters jointly with the task model being deployed. In this scheme, the ISTN would be optimized to maximize the performance of the predictor on the transformed images $S2T$. However, if the trained ISTN is then used to train models for other tasks, it is not guaranteed to reach the best performance.

This work takes another approach. The ISTN is trained independently of the task model such that updates to the ISTN parameters do not depend on the predictor output. The ISTN updates must rely only on a domain discriminator and its internal losses. The predictor model uses only the output of the ISTN to reduce its error. In this scheme, the ISTN should converge to a solution which is the same regardless of the predictor model being trained. The ISTN could be trained until the performance of the classifier *evaluated on the ISTN output* reaches a maximum. The same ISTN would then be used to transform images which can be used in training a different task model. This will be referred to as the task-agnostic training approach.

This work validates the use of ISTNs for *task-agnostic unsupervised* DA on two problems: (1) a proof-of-concept showing recovery of a classifier's performance on digit recognition; and (2) classification and regression tasks with real-world, multi-site T1-weighted brain MRI.

5.4.2 Image and Spatial Transformer Networks (ISTNs).

ISTNs have two components: an Image Transformer Network (ITN) and a Spatial Transformer Network (STN) [82, 99]. The output from the ITN is passed directly to the input of the STN. When the two networks are trained end-to-end they can be considered as single unit (ISTN). This is the approach taken in the original work [99] though it is also possible to optimize the parameters of the two networks separately. This work follows the original approach.

5.4.2.1 Image Transformer Networks (ITNs).

The ITN performs image-level appearance transformations such as contrast and brightness changes. These are learned from the target distribution during training. ITNs are I2I translation networks [81, 184]: they take an image as input and their output is also an image. I2I networks are typically based on encoder-decoder architectures, for example the U-Net [137] with residual skip connections can be employed. The encoder performs convolutions on the input and reduces its size down to some 'bottleneck' where further convolutions are done without modifying the input shape. This reduces the image-level representation of the input down to a much smaller latent representation.

To create an output image from this latent representation, the decoder of the ITN must restore the dimensions of the image. This is achieved by reversing the convolution process with transpose convolutions or up-convolutions. It has been well-observed that traditional transpose convolutional approaches result in the appearance of chequerboard-like artifacts in the output image [115]. The effect is particularly noticeable in areas of uniform intensity *e.g.* backgrounds. To combat this

undesirable effect, the latent representation can first be upsampled to the required size and then a convolution with stride of 1 applied.

In this work, a U-Net-based I2I translation model is employed. The output at each level of the decoder is concatenated with the output featuremaps from the corresponding level of the encoder. Batch-normalization (BN) is applied after each convolutional layer on the encoder pathway. Dropout is then applied and the output passed through a Rectified Linear Unit (ReLU). A final $\tanh$ activation is applied to the output. The resulting output image has the same dimensions as the input. The specific ISTN architectures employed in this work are given in their respective sections: table 5.6 on page 157 and table 5.9 on page 160.

5.4.2.2 Spatial Transformer Networks (STNs)

ITNs learn changes that are limited to the global and localised luminance and contrast of the input image. This largely leaves the structure of the image unchanged. However, if the expected behaviour of a model is to perform explicit spatial transformations on the input, STNs [82] can be used.

In this work, global and local spatial transformations are considered. Explicitly, we apply global translation, rotation, scaling and shearing operations at the image level to achieve global spatial transformations. These are affine transformations. Local spatial transformations are achieved by creating a mesh grid of control-points in the image space and learning B-Spline kernel transformations which are used to warp the image.

Affine transformations. Global affine transformations are performed by the matrix multiplication of the image matrix and an affine transformation matrix as detailed in section 2.4.1.4 on page 77. The affine STN consists of a Convolutional Neural Network (CNN) whose output is passed to four separate regressors which learn the parameters of the linear spatial transforms: translation, rotation, scaling and shearing. Parameters are learned for the transformations in each spatial dimension, with the exception of a single rotation angle and a single shear angle in the 2D case. The transformations can be limited to exclude extreme modifications.

B-Spline Transformations. The input to the B-spline STNs is passed through several strided covolutional layers each followed by BN. The output from the convolutional block is passed to a Fully-connected (FC) layer acting as a regressor which learns Control Point (CP) displacements from the regular grid in each dimension. Displacements are defined as a proportion of the image size by forcing them into the range $[-1, 1]$. Displacements are capped by a user defined maximum displacement variable. The input image is resampled with linear interpolation using a cubic (order $n = 3$) B-spline filter. The filter is constructed by $n = 3$ recursive convolutions of the 0^{th} order B-spline basis function.

Table 5.1: Combinations of ITN and STNs Considered in the DA Experiments. Image Transformer Network (ITN) pass their outputs to the input one of two types of Spatial Transformer Network (STN). Not all combinations of Image and Spatial Transformer Network (ISTN) are considered for each dataset. *M*: Morpho-MNIST dataset. *B*: Brain MRI dataset. Note that no ITN and no STN would mean no DA.

	ITN	
STN	✗	✓
None	-	M & B
Affine	M	M & B
B-Spline	M	M & B

The combinations of ITN and STN used in this work are shown in table 5.1. In the medical imaging case, the 'STN-only' combination is not considered. This is for two reasons: first, it is clear that luminance and contrast changes, more than structural and spatial changes, are the primary causes of DS between multi-site datasets using the same modality. Second, the STNs are more complex and take longer to train than the ITNs especially in the 3D medical imaging case. Thus to save computational time and maintain focus on the research problem, the 'STN-only' scenario is used solely in demonstrating the application of ISTNs in the proof-of-concept experiments.

5.4.3 Training.

Adversarial training is the scheme used to train Generative Adversarial Networks (GANs) and I2I transformer networks. In adversarial training, the generator model tries to produce increasingly convincing fakes which fool an improving discriminator. In this work, the ISTN must produce images in the target domain which confuse a domain discriminator. The discriminator and ISTN are trained using separate optimizers.

5.4.3.1 Discriminator Training.

The source domain images S are passed through the ISTN to generate images $S2T$, where T indicates images from the target domain. Next, $S2T$ are passed through the discriminator D_T to yield a score in the range $(0, 1)$ denoting whether the image is a real sample from domain T or a transformed one. The discriminator is trained by minimizing the binary cross-entropy loss $\mathcal{L}_{bce}$ between the predicted and true domain labels. Equation (5.2) on the following page shows the total discriminator loss for D_T. A similar loss is shown in equation (5.3) on the next page for D_S which determines whether an image is a real sample from the source domain or a transformed

one.

Discriminator Loss.

$$\mathcal{L}_{disT} = \frac{1}{2}\left[\mathcal{L}_{bce}(D_T(S2T), 0) + \mathcal{L}_{bce}(D_T(T), 1)\right]. \tag{5.2}$$

$$\mathcal{L}_{disS} = \frac{1}{2}\left[\mathcal{L}_{bce}(D_S(S2T), 1) + \mathcal{L}_{bce}(D_S(T), 0)\right]. \tag{5.3}$$

It is common to use soft labels [111] for the true domain to stabilize early training of the discriminator. The hard '0' and '1' domain labels are replaced by random uniform values in the ranges $[0.00,\ 0.03]$ and $[0.97,\ 1.00]$, respectively. By training a strong domain discriminator, we force the ISTN to become better at producing images in the target domain.

5.4.3.2 ISTN Training.

The combined ISTN is trained end-to-end by an optimizer which finds the parameters of the networks that best minimize an objective function. The objective function is composed of a number of individual loss functions which are described here. The framework is an extension of the CycleGAN [184] approach laid out in section 2.2.4 on page 49.

A Identity Loss.

When input images S from the source domain $\mathbf{S}$ are passed through the ISTN I_{S2T} they are transformed into the target domain $\mathbf{T}$. When I_{S2T} receives images T that are already in the target domain $\mathbf{T}$, it should apply the identity transformation so the output images $T2T$ remain unchanged. To achieve this, the ℓ_1 norm between T and the transformed images $T2T$ is minimized. This 'identity loss' $\mathcal{L}_{idt}$ is defined in equation (5.4).

Identity Loss.

$$\mathcal{L}_{idt} = \|I_{S2T}(T) - T\|_1 \tag{5.4}$$

$$= \|T2T - T\|_1. \tag{5.5}$$

B Cycle Consistency Loss.

This loss was first introduced with the CycleGAN [184]. The bidirectional I2I translation method uses two ISTNs: I_{S2T} for transformations of images from the source $\mathbf{S}$ to target $\mathbf{T}$ domain and I_{T2S} for transformations from $\mathbf{T}$ to $\mathbf{S}$. Images S and T are transformed to $S2T$ and $T2S$ when passed through I_{S2T} and I_{T2S} respectively. The cycle consistency loss enforces the constraint that when $S2T$ pass through I_{T2S} then the resulting images $S2T2S$ should be very close to the original S. Similarly, when $T2S$ pass through I_{S2T}, the ISTN should output images $T2S2T$ which are very close to T. The cycle consistency loss $\mathcal{L}_{cyc}$ is the sum of the ℓ_1 norms between input and output images shown in equation (5.6).

Cycle Consistency Loss.

$$\mathcal{L}_{cyc} = \frac{1}{2}\left[\|S2T2S - S\|_1 + \|T2S2T - T\|_1\right] \tag{5.6}$$

C Adversarial Loss.

With only $\mathcal{L}_{idt}$ and $\mathcal{L}_{cyc}$ the function learned by the ISTN would tend towards the identity function. It is the relationship between the discriminator and ISTN that forces the ISTN to learn something about the target domain.

The ISTN output $S2T$ is passed through the discriminator and is forced to be closer to domain T by computing the adversarial loss $\mathcal{L}_{adv}$ in equation (5.7). The hard '1' domain label is replaced by random uniform values in the range $[0.97,\ 1.00]$ to stabilize training in the early stages.

Adversarial Loss.
$$\mathcal{L}_{adv} = \frac{1}{2}\left[\mathcal{L}_{bce}(D_T(S2T), 1) + \mathcal{L}_{bce}(D_S(T2S), 1)\right] \tag{5.7}$$

This work compares two methods of adversarial training: unidirectional and bidirectional. In the unidirectional case, we train a single ISTN I_{S2T} to transfer images from the source $\mathbf{S}$ to the target $\mathbf{T}$ domain. The bidirectional case uses the CycleGAN formulation where two ISTNs are trained simultaneously: I_{S2T} to transfer images from the source $\mathbf{S}$ to the target $\mathbf{T}$ domain and I_{T2S} for the reverse.

The cycle consistency loss $\mathcal{L}_{cyc}$ is used only in the bidirectional case as it requires an ISTN for each direction; it is omitted in the unidirectional case. Additionally, two discriminators are required in the bidirectional case: one for the source domain D_S and one for the target domain D_T. Figure 5.1 on the following page shows the two ISTN training schemes, their outputs and associated losses.

In both cases, a trade-off or weighting is applied to the individual losses that comprise the overall objective. A weighting factor λ is used to control the relative importance of $\mathcal{L}_{idt}$ and $\mathcal{L}_{cyc}$ with $\mathcal{L}_{idt}$ considered to be half as important as $\mathcal{L}_{cyc}$.

Unidirectional Training. The complete objective function for the ISTN which transfers images from source $\mathbf{S}$ to target $\mathbf{T}$ domains is shown in equation (5.8). In this scheme, there is no cycle-consistency loss, however λ is still applied and the half retained to avoid redefinition of the variable in the bidirectional training scenario.

Unidirectional Loss.
$$\mathcal{L}_{istn}^{S2T} = \mathcal{L}_{adv}^{S2T} + \tfrac{1}{2}\lambda\mathcal{L}_{idt}^{S2T} \tag{5.8}$$
$$= \mathcal{L}_{bce}(D_T(S2T), 1) + \tfrac{1}{2}\lambda \left\|T2T - T\right\|_1 . \tag{5.9}$$

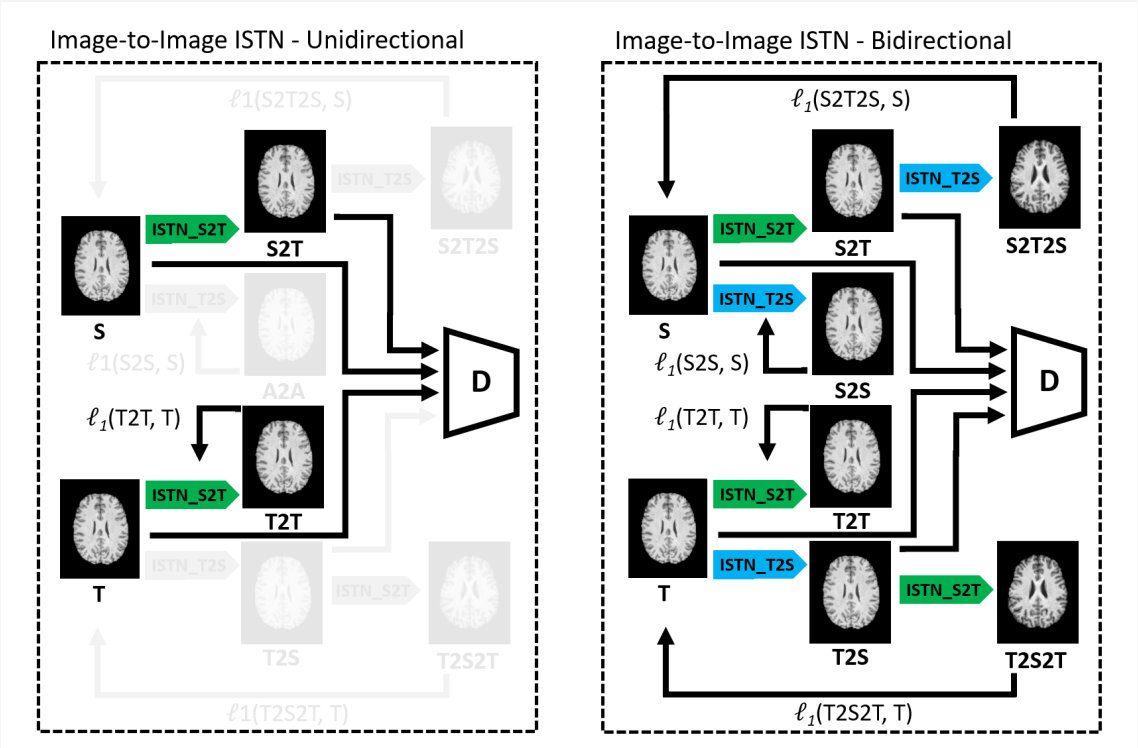


Figure 5.1: Comparison of Uni- and Bi-directional ISTN Adversarial Training Approaches (left) The ISTN can be trained adversarially such that the discriminator D becomes better at identifying real (S and T) and transformed ($S2T$) images. The ISTN simultaneously produces better transformations $S2T$ of S that look more like the images T. The training process can also be done bidirectionally (right) which is an extension of the CycleGAN framework [184]. Grayed out components (left) show that the unidirectional training scheme uses a subset of the bidirectional components.

Bidirectional Training. The complete objective function for ISTNs in the bidirectional case is shown in equation (5.10). The individual objectives for I_{S2T} and I_{T2S} are also shown.

$$\mathcal{L}_{istn}^{Bi} = \mathcal{L}_{adv}^{Bi} + \tfrac{1}{2}\lambda\mathcal{L}_{idt}^{Bi} + \lambda\mathcal{L}_{cyc} \tag{5.10}$$

$$\mathcal{L}_{adv}^{Bi} = \tfrac{1}{2}\left[\mathcal{L}_{adv}^{S2T} + \mathcal{L}_{adv}^{T2S}\right] = \tfrac{1}{2}\left[\mathcal{L}_{bce}(D_T(S2T), 1) + \mathcal{L}_{bce}(D_S(T2S), 1)\right] \tag{5.11}$$

$$\mathcal{L}_{idt}^{Bi} = \tfrac{1}{2}\left[\mathcal{L}_{idt}^{S2T} + \mathcal{L}_{idt}^{T2S}\right] = \tfrac{1}{2}\left[\|S2S - S\|_1 + \|T2T - T\|_1\right] \tag{5.12}$$

$$\mathcal{L}_{cyc} = \tfrac{1}{2}\left[\|S2T2S - S\|_1 + \|T2S2T - T\|_1\right] \tag{5.13}$$

In bidirectional training, the total loss for the discriminators is combined as shown in equation (5.14).

$$\mathcal{L}_{dis} = \tfrac{1}{2}\left[\mathcal{L}_{dis}^{S} + \mathcal{L}_{dis}^{T}\right] \tag{5.14}$$

$$\mathcal{L}_{dis}^{S} = \tfrac{1}{2}\left[\mathcal{L}_{bce}(D_S(T2S), 0) + \mathcal{L}_{bce}(D_S(S), 1)\right] \tag{5.15}$$

$$\mathcal{L}_{dis}^{T} = \tfrac{1}{2}\left[\mathcal{L}_{bce}(D_T(S2T), 0) + \mathcal{L}_{bce}(D_T(T), 1)\right] \tag{5.16}$$

Table 5.2: Parameters for each Morpho-MNIST dataset used in DA experiments The four datasets are (left-to-right) (A) thin-unslanted; (B) thick-unslanted; (C) thin-slanted; and (D) thick-slanted.

Domain	A	B	C	D
Training n	60,000	60,000	60,000	60,000
Validation n	10,000	10,000	10,000	10,000
Thickness (pixels)	2.5	2.5	5.0	5.0
Shear (°)	0	20-25	0	20-25

5.4.4 Datasets.

Two types of data are analyzed in this work: small, grayscale 2D images of handwritten digits; and large, single-channel, 3D brain MRI. This section describes the data sources and their processing.

5.4.4.1 MorphoMNIST.

The well-known MNIST [98] dataset contains 70,000 2-dimensional grayscale images of hand-drawn digits each 28×28 pixels in size. Morpho-MNIST [24] is a framework that enables the application of medically-inspired perturbations, such as local swellings and fractures, to the MNIST dataset. The framework contains methods to measure and modify the thickness and shear of the original MNIST digits. In this work, four Morpho-MNIST datasets are created.

A **Thin-unslanted dataset.** Each digit in the MNIST dataset is rectified such that the shear, measured with Morpho-MNIST is $0°$. The digit is then modified to be 2.5 pixels in thickness. Thus the digits are all 'thin' and 'unslanted'. This dataset is referred to as domain $\mathbf{A}$.

B **Thick-unslanted dataset.** Each digit in the MNIST dataset is rectified such that the shear, measured with Morpho-MNIST is $0°$. The digit is then modified to be 5.0 pixels in thickness. Thus the digits are all 'thick' and 'unslanted'. This dataset is referred to as domain $\mathbf{B}$.

C **Thin-slanted dataset.** Each digit in the MNIST dataset is first measured with Morpho-MNIST to determine its shear angle. A shear transformation is applied to the digit such that its *overall* shear is between 20–25°. The digit is then modified to be 2.5 pixels in thickness. Thus the digits are all 'thin' and 'slanted'. This dataset is referred to as domain $\mathbf{C}$.

D **Thick-slanted dataset.** Each digit in the MNIST dataset is first measured with Morpho-MNIST to determine its shear angle. A shear transformation is applied to the digit such that its *overall* shear is between 20–25°. The digit is then modified to be 5.0 pixels in thickness. Thus the digits are all 'thick' and 'slanted'. This dataset is referred to as domain $\mathbf{D}$.

The datasets are summarised in table 5.2. Each sample is zero-padded such that the final image

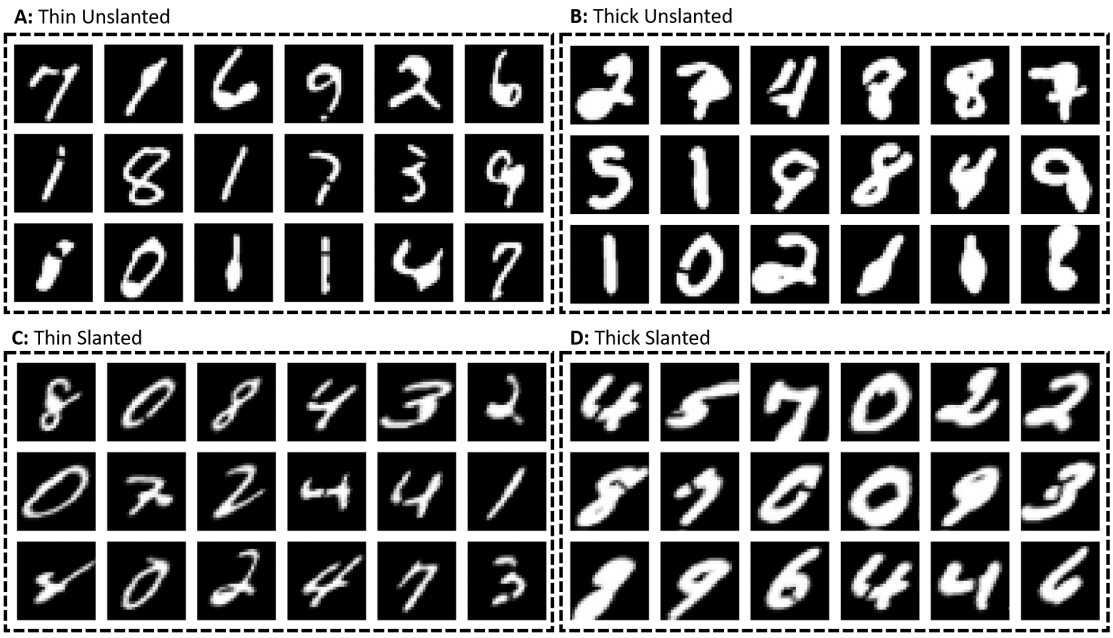


Figure 5.2: Samples from Each of 4 Constructed Morpho-MNIST Datasets. (top-to-bottom) $\mathbf{A}$: 'thin-unslanted' digits; $\mathbf{B}$ 'thick-unslanted' digits; $\mathbf{C}$ 'thin-slanted' digits and $\mathbf{D}$ 'thick-slanted'. Each domain contains 3 classes: 'fractured', 'swollen' and 'healthy' digits.

dimensions are 32×32. Each digit in $\mathbf{A} - \mathbf{D}$ is assigned a random class: 'healthy', 'fracture' or 'tumor'. For the fracture class, the Morpho-MNIST framework is used to apply a single pixel-width fracture to the digit. Digits assigned to the tumor class are 'swollen' at a random point along the digit. This is done by applying a radial power transform ($\gamma = 2.5$) within a radius of 6–7.5 pixels from the randomly chosen swelling center. The healthy class has no perturbation applied. All digits in the four datasets $\mathbf{A} - \mathbf{D}$ are unique in the sense that they are each assigned a random class, shear angle and thickness. Examples from each dataset are shown in 5.2.

5.4.4.2 Brain Magnetic Resonance Images.

Two datasets are created for this work using images from independent imaging studies which both acquire T1-weighted Brain MRI: the UK Biobank (UKBB) and the Cambridge Centre for Ageing and Neuroscience (Cam-CAN).

The UK Biobank (UKBB) Imaging Study. The UKBB [4, 108, 154] was described in section 3.4.0.1 on page 96. In summary, it is a registered charity founded in 2006 which recruited 500,000 participants to undergo a battery of analyses with the aim for 100,000 participants to have an array of imaging procedures. Each subject has a raft of brain scans including the T1-weighted MRI that is used in this work. All UKBB images are acquired at UKBB imaging centres in the UK[1].

Table 5.3: Acquisition Parameters for the Multi-site Brain MRI Datasets. The UKBB and Cam-CAN studies acquire brain structural MRI images using similar parameters: 1 mm isotropic images using the 3D MPRAGE pulse sequence on Siemens 3 T scanners with a 32-channel receiver head coil and in-plane acceleration factor of 2. The parameters that differ are shown here including Repetition Time (TR), Echo Time (TE), Inversion Time (TI), Acquisition Time (TA) and Field-of-view (FOV)

Site	Scanner	TR (ms)	TE (ms)	TI (ms)	TA (s)	FOV (mm)
Cam-CAN	Siemens TIM Trio	2250	2.99	900	272	256 x 240 x 192
UKBB	Siemens Skyra	2000	2.01	880	294	208 x 256 x 256

The Cambridge Centre for Ageing and Neuroscience (Cam-CAN). Cam-CAN [148, 161] was launched in 2010 and "uses epidemiological, cognitive, and neuroimaging data to understand how individuals can best retain cognitive abilities into old age"[2]. The study acquires structural MRI images alongside resting and task-based functional MRI and Magnetoencephalography (MEG) data. Behavioural experiments are also performed outside of the scanner. All Cam-CAN images are acquired at the Medical Research Council (MRC) Cognition and Brain Sciences Unit, Cambridge, UK.

Both the Cam-CAN and UKBB studies acquire 1 mm isotropic images using the 3D MPRAGE pulse sequence on Siemens 3T scanners with a 32-channel receiver head coil and in-plane acceleration factor of 2. Table 5.3 presents the acquisition parameters that differ between the two sites. The acquisition parameters of both sites are mostly similar, and the images cannot be easily distinguished on visual inspection.

Two datasets of T1-weighted brain MRI are created. The first dataset contains $n = 689$ subjects from the UKBB imaging study. The UKBB dataset in this work contains equal numbers of male and female subjects between the ages of 48 and 71 ($\mu = 59.5$). All subjects have no reported pathology.

The second dataset comprises $n = 565$ subjects from the Cam-CAN study. This dataset has a greater age range (30–87, $\mu = 57.9$) than the UKBB dataset but maintains the male-to-female ratio. All subjects have no reported pathology.

For pre-processing, all images are aligned to Montreal Neurological Institute (MNI) space [29] using only affine transformations, *i.e.* translation, rotation, scaling and shearing, as described in section 2.4.1.4 on page 77. All images are skull-stripped; bias-field-corrected; and intensity-normalised within a brain mask to zero mean and unit variance. Voxels outside the mask are set to 0 intensity. Images are passed through a $\tanh$ function before being consumed by the networks which scales their intensity distributions to the range $(-1, 1)$.

Table 5.4: Architecture of 3-class Classifier for Morpho-MNIST Datasets nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization (BN: batch normalization), D: Dropout keep-rate, A: activation function.

3-Class Classifier Architecture - Morpho-MNIST

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	16	3	1	1	[1, 24, 24]	[16, 24, 24]	-	-	ReLU
conv	32	3	2	1	[16, 14, 14]	[32, 7, 7]	BN	-	ReLU
conv	64	3	2	1	[32, 7, 7]	[64, 4, 4]	BN	-	ReLU
conv	128	3	2	1	[64, 4, 4]	[128, 1, 1]	BN	0.5	ReLU
conv	3	3	2	0	[128, 1, 1]	[3, 1, 1]	-	-	sigmoid
out	-	-	-		-	[3, 1, 1]	-	-	-

5.4.5 Task Models.

This work investigates an approach to minimize the DS problem. It measures the change in performance of task models when they are retrained on transformed data and evaluated on the new domain. Three experiments are performed: a 3-class disease classification problem on the different Morpho-MNIST domains; a sex-classification problem on brain MRI; and an age regression problem on brain MRI.

5.4.5.1 Task: Morpho-MNIST Disease Classification.

The downstream task in this experiment is a 3-class classification problem: 'healthy' vs. 'fractured' vs. 'swollen'. First, a small, fully-convolutional classifier $\mathcal{T}_A$ is trained to perform the classification on domain $\mathbf{A}$ - the 'thin-unslanted' dataset. $\mathcal{T}_A(B)$, $\mathcal{T}_A(C)$ and $\mathcal{T}_A(D)$ are evaluated on the other domains $\mathbf{B} - \mathbf{D}$. ISTNs are used to retrain $\mathcal{T}_A$ on transformed images $A2B$, $A2C$, and $A2D$, which is then evaluated on the corresponding test domains $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$.

The ISTN is trained for 100 epochs and grid search is performed to find suitable hyper-parameters including learning rate, loss term weighting λ and the CP spacing of the B-spline STN. Experiments are conducted using ITN only, STN only and combinations of affine and B-spline ISTNs to determine the best model for the task (see table 5.1 on page 148).

The specific architectures for the classifier, discriminator and ISTN are laid out in table 5.4, table 5.5, and table 5.6 respectively.

Table 5.5: ISTN Discriminator Architecture for Morpho-MNIST Experiments. nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization (IN: instance normalization), D: Dropout keep-rate, A: activation function.

Discriminator Architecture - Morpho-MNIST

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 28, 28]	-	-		-
conv	32	3	1	1	[1,28,28]	[32,28,28]	-	-	ReLU
conv	64	3	2	1	[32,28,28]	[64,14,14]	IN	-	ReLU
conv	128	3	2	1	[64,14,14]	[128,7,7]	IN	-	ReLU
conv	256	3	2	1	[128,7,7]	[256,4,4]	IN	0.5	ReLU
conv	1	3	2	1	[256,4,4]	[1,1,1]	-	-	sigmoid
out	-	-	-		-	[1,1,1]	-	-	-

Table 5.6: ISTN architectures employed in Morpho-MNIST experiments Architecture of the ITN (top) and STNs (bottom) for Morpho-MNIST classification experiment. nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization (BN: batch normalization, IN: instance normalization), D: Dropout keep-rate, A: activation function.'up' is composed of bilinear upsampling followed by zero-padding of 1 and convolution shown in the table. '*' indicates where inputs are shared across the regressors for translation[1], rotation[2], scaling[3] and shearing[4] affine matrix parameters. N_{cp} is the number of control-point in the B-Spline transform grid.

ITN Architecture - Morpho-MNIST

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 28, 28]	-	-		-
conv	16	3	1	1	[1, 28, 28]	[16, 28, 28]	BN	-	ReLU
conv	32	3	2	1	[16, 28, 28]	[32, 14, 14]	BN	-	ReLU
conv	64	3	2	1	[32, 14, 14]	[64, 7, 7]	BN	-	ReLU
conv	128	3	1	1	[64, 7, 7]	[128, 7, 7]	BN	-	ReLU
conv	64	3	1	1	[128, 7, 7]	[64, 7, 7]	BN	-	ReLU
up	32	3	1	1	[64, 7, 7]	[32, 14, 14]	BN	-	ReLU
up	16	3	1	1	[32, 14, 14]	[16, 28, 28]	BN	-	ReLU
up	1	3	1	1	[16, 28, 28]	[1, 28, 28]	-	-	tanh
out	-	-	-	-	[1, 28, 28]	-	-	-	-

Affine STN Architecture - Morpho-MNIST

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 32, 32]	-	-		-
conv	8	3	1	1	[1, 32, 32]	[8, 32, 32]	BN	-	ReLU
conv	16	3	2	1	[8, 32, 32]	[16, 16, 16]	BN	-	ReLU
conv	32	3	2	1	[16, 16, 16]	[32, 8, 8]	BN	-	ReLU
conv	64	3	2	1	[32, 8, 8]	[64, 4, 4]	BN	-	ReLU
conv	64	3	2	1	[128, 8, 8]	[128, 2, 2]*	BN	-	ReLU
linear[1]	2	-	-	-	$[128 \times 2 \times 2]$*	[2]	-	-	-
linear[2]	1	-	-	-	$[128 \times 2 \times 2]$*	[1]	-	-	-
linear[3]	2	-	-	-	$[128 \times 2 \times 2]$*	[2]	-	-	-
linear[4]	1	-	-	-	$[128 \times 2 \times 2]$*	[1]	-	-	-

B-Spline STN Architecture - Morpho-MNIST

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 32, 32]	-	-		-
conv	4	5	1	2	[1, 32, 32]	[4, 32, 32]	BN	-	ReLU
conv	8	5	2	2	[4, 32, 32]	[8, 16, 16]	BN	-	ReLU
conv	16	5	2	2	[8, 16, 16]	[16, 8, 8]	BN	-	ReLU
conv	32	5	2	2	[16, 8, 8]	[32, 4, 4]	BN	-	ReLU
conv	64	5	2	2	[32, 8, 8]	[64, 2, 2]	BN	-	ReLU
linear	N_{cp}	-	-	-	$[64 \times 2 \times 2]$	$[N_{cp}]$	-	-	-

Table 5.7: Architecture of Sex Classifier for Brain MRI. nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization (BN: batch normalization), D: Dropout keep-rate, A: activation function.

Sex Classifier Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	8	5	2	2	[1, 64, 64, 64]	[8, 64, 64, 64]	-	-	ReLU
conv	16	5	2	2	[8, 64, 64, 64]	[16, 32, 32, 32]	BN	-	ReLU
conv	32	5	2	2	[16, 32, 32, 32]	[32, 16, 16, 16]	BN	-	ReLU
conv	64	5	2	2	[32, 16, 16]	[64, 8, 8]	BN	0.5	ReLU
conv	128	5	2	2	[64, 8, 8]	[128, 4, 4]	BN	0.5	ReLU
conv	128	5	2	2	[128, 4, 4]	[128, 1, 1]	BN	0.5	ReLU
conv	1	5	1	2	[128, 1, 1]	[1, 1, 1]	-	-	sigmoid
out	-	-	-		-	[1, 1, 1]	-	-	-

5.4.5.2 Task: Brain MRI Sex Classification.

In this experiment, a small, full-convolutional classifier $\mathcal{T}_U$ is trained to perform sex classification on the the UKBB domain $\mathbf{U}$ and evaluated on the Cam-CAN domain $\mathbf{C}$. The classifier is trained for 100 epochs with learning rate 0.0001.

In the DS experiments, the classifier is retrained on the transformed images $U2C$ produced by the ISTN and evaluated on the real images from the corresponding target domain. Both transfer directions are considered by switching the roles of source and target domains. From each site, 450 subjects are used for training and the remainder for testing.

In this experiment, the age-range of the Cam-CAN dataset is much larger than that of the UKBB. This simulates population shift between the datasets which compounds the already present acquisition shift. This experiment aims to show recovery of the sex classifier from both types of DS.

The ISTN is trained for 100 epochs and grid search is performed to find suitable hyper-parameters including learning rate, loss term weighting λ and the CP spacing of the B-spline STN. As laid out in table 5.1 on page 148, experiments are conducted using ITN only, and combinations of affine and B-spline ISTNs to determine the best model for the task.

The specific architectures for the classifier, discriminator, and ISTN are laid out in table 5.7, table 5.8 on the next page, and table 5.9 on page 160 respectively.

5.4.5.3 Task: Brain MRI Age Regression.

A small LeNet-based [98] regressor is trained to perform age regression on the UKBB domain $\mathbf{U}$ and is evaluated on the Cam-CAN domain $\mathbf{C}$. The model comprises two convolution-MaxPool-ReLU blocks and four FC layers. The regressor is trained for 100 epochs with a learning rate of 0.0001.

Table 5.8: ISTN Discriminator Architecture for Brain MRI Experiments. nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization (IN: instance normalization), D: Dropout keep-rate, A: activation function.

Discriminator Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	32	3	1	1	[1, 64, 64, 64]	[32, 64, 64, 64]	-	-	ReLU
conv	64	3	2	1	[32, 64, 64, 64]	[64, 32, 32, 32]	IN	-	ReLU
conv	128	3	2	1	[64, 32, 32, 32]	[128, 16, 16, 16]	IN	-	ReLU
conv	256	3	2	1	[128, 16, 16]	[256, 8, 8]	IN	-	ReLU
conv	256	3	2	1	[256, 8, 8]	[256, 4, 4]	IN	0.5	ReLU
conv	1	3	2	1	[256, 4, 4]	[1, 1, 1]	-	-	sigmoid
out	-	-	-		-	[1, 1, 1]	-	-	-

In the DS experiments, the regressor is retrained on the transformed images $U2C$ produced by the ISTN and evaluated on the real images from the corresponding target domain. Both transfer directions are considered by switching the roles of source and target domains.

The age range of both datasets in the regression task is matched to reduce the population shift, limiting the DS only to the more subtle scanner effects. The purpose is to investigate whether DA with ISTNs is able to recover smaller changes in model performance.

The ISTN is trained for 200 epochs and grid search is performed to find suitable hyper-parameters including learning rate, loss term weighting λ and the CP spacing of the B-spline STN. As laid out in table 5.1 on page 148, experiments are conducted using ITN only, and combinations of affine and B-spline ISTNs to determine the best model for the task.

The architectures for the ISTNs and discriminator remain the same as in the sex classification experiment. The architecture for the age regressor is shown in table 5.10 on page 161.

Table 5.9: ISTN Architecture for Brain MRI Experiments. ITN (top) and STN (bottom) architecture for Brain MRI sex classification and age regression experiments. nf: number of channels, k: square kernel size, s: stride in and out: layer input and output dimensions, N: normalization (BN: batch normalization), D: Dropout keep-rate, A: activation function. 'up' is composed of trilinear upsampling followed by zero-padding of 1 and convolution shown in the table. '*' indicates where inputs are shared across the regressors for translation[1], rotation[2], scaling[3] and shearing[4] affine matrix parameters. N_{cp} is the number of control-point in the B-Spline transform grid.

ITN Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	8	3	1	1	[1, 64, 64, 64]	[8, 64, 64, 64]	BN	-	ReLU
conv	16	3	2	1	[8, 64, 64, 64]	[16, 32, 32, 32]	BN	-	ReLU
conv	32	3	2	1	[16, 32, 32, 32]	[32, 16, 16, 16]	BN	-	ReLU
conv	64	3	2	1	[32, 16, 16, 16]	[64, 8, 8, 8]	BN	-	ReLU
conv	64	3	1	1	[64, 8, 8, 8]	[64, 8, 8, 8]	BN	-	ReLU
conv	64	3	1	1	[64, 8, 8, 8]	[64, 8, 8, 8]	BN	-	ReLU
up	32	3	1	1	[64, 8, 8, 8]	[32, 16, 16, 16]	BN	0.5	ReLU
up	16	3	1	1	[32, 16, 16, 16]	[16, 32, 32, 32]	BN	0.5	ReLU
up	8	3	1	1	[16, 32, 32, 32]	[8, 64, 64, 64]	BN	0.5	ReLU
up	1	3	1	1	[8, 64, 64, 64]	[1, 64, 64, 64]	-	-	tanh
out	-	-	-	-	-	[1, 64, 64, 64]	-	-	-

Affine STN Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	8	3	1	1	[1, 64, 64, 64]	[8, 64, 64, 64]	BN	-	ReLU
conv	16	3	2	1	[8, 64, 64, 64]	[16, 32, 32, 32]	BN	-	ReLU
conv	32	3	2	1	[16, 32, 32, 32]	[32, 16, 16, 16]	BN	-	ReLU
conv	32	3	2	1	[32, 16, 16, 16]	[32, 8, 8, 8]	-	-	ReLU
linear[1]	3	-	-	-	$[128 \times 8 \times 8 \times 8]$*	[3]	-	-	-
linear[2]	3	-	-	-	$[128 \times 8 \times 8 \times 8]$*	[3]	-	-	-
linear[3]	3	-	-	-	$[128 \times 8 \times 8 \times 8]$*	[3]	-	-	-
linear[3]	3	-	-	-	$[128 \times 8 \times 8 \times 8]$*	[3]	-	-	-

B-Spline STN Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	8	3	1	1	[1, 64, 64, 64]	[8, 64, 64, 64]	BN	-	ReLU
conv	16	3	2	1	[8, 64, 64, 64]	[16, 32, 32, 32]	BN	-	ReLU
conv	32	3	2	1	[16, 32, 32, 32]	[32, 16, 16, 16]	BN	-	ReLU
conv	64	3	2	1	[32, 16, 16, 16]	[64, 8, 8, 8]	-	-	ReLU
linear	N_{cp}	-	-	-	$[64 \times 8 \times 8 \times 8]$	$[N_{cp}]$	-	-	-

Table 5.10: Age Regressor Architecture for Brain MRI. nf: number of channels, k: square kernel size, s: stride, in and out: layer input and output dimensions, N: normalization, D: Dropout keep-rate, A: activation function.

Age Regressor Architecture - Brain MRI

layer	nf	k	s	p	in	out	N	D	A
in	-	-	-	-	[1, 64, 64, 64]	-	-		-
conv	16	3	1	1	[1, 64, 64, 64]	[8, 64, 64, 64]	-	-	ReLU
maxpool	-	-	2	1	[8, 64, 64, 64]	[8, 32, 32, 32]	-	-	-
conv	32	3	2	1	[8, 32, 32, 32]	[32, 32, 32, 32]	-	-	ReLU
maxpool	-	-	2	1	[32, 32, 32, 32]	[32, 16, 16, 16]	-	-	-
linear	128	-	-	-	$[32 \times 16 \times 16 \times 16]$	[128]	-	-	ReLU
linear	64	-	-	-	[128]	[64]	-	-	ReLU
linear	32	-	-	-	[64]	[32]	-	-	ReLU
linear	1	-	-	-	[32]	[1]	-	-	
out	-	-	-		-	[1]	-	-	-

5.5 Results

For each dataset, the performance of the task model is first assessed before evaluating the the ISTN experiments.

5.5.1 Morpho-MNIST: 3-class Disease Classification Results

The 3-class 'disease' classifier outlined in section 5.4.5.1 on page 155 was trained on the 'thin-unslanted' digit domain $\mathbf{A}$ for 500 epochs and learning rate 0.0001. Loss and accuracy progression over the training period is shown in figure 5.3 on the following page. This task model is denoted $\mathcal{T}_A$. The best performance of $\mathcal{T}_A$ is $\mathcal{T}_A(A) = 0.9579$, a classification accuracy of 96%.

To demonstrate the effect of DS on the performance of $\mathcal{T}_A$, it is evaluated on the remaining domains $\mathbf{B} - \mathbf{D}$. These baselines are shown in table 5.11 on the next page. The performance of $\mathcal{T}_A$ is significantly degraded when evaluated on the unseen domains. This highlights the DS problem.

5.5.2 Morpho-MNIST: DA with ISTN Results

In the first experiment, domain $\mathbf{A}$ was set as the source domain $\mathbf{S}$ and $\mathbf{B}$ was assigned as the target domain $\mathbf{T}$. The classifier trained on the source domain is $\mathcal{T}_S$.

Each ISTN $\mathcal{I}$ in table 5.1 on page 148 is trained to transfer images from the source $\mathbf{S}$ to target $\mathbf{T}$ domains. A classifier $\mathcal{T}_{S2T}$ is trained simultaneously either from scratch or finetuned from $\mathcal{T}_S$. Both $\mathcal{I}$ and $\mathcal{T}_{S2T}$ are trained until the performance $\mathcal{T}_{S2T}(S2T)$ reaches its maximum. Quantitative results for each experiment and ISTN combination are shown in table 5.12 on page 164.

The transfer from the *thin*-unslanted domain to the *thick*-unslanted domain requires an intensity transformation of the image around the original strokes of the digit. Results show that training a new classifier $\mathcal{T}$ from scratch on the ISTN output $A2B$ is able to recover the performance from $\mathcal{T}_A(B) = 41.2\,\%$ to $\mathcal{T}_{A2B}(B) = 79.0\,\%$ when using an ITN without an STN. Fine-tuning the original classifier $\mathcal{T}_A$, rather than training from scratch, further improves the performance $\mathcal{T}_{A2B}(B) = 83.3\,\%$. When the STN is employed without an ITN, the performance is only partially recovered. This indicates that the STN component of the ISTN does not contribute much to the transformations in this case. The affine STN, both acting alone and in tandem with the ITN, appears to recover more of the model's performance than when B-spline STNs are used. Perhaps the scaling transformation may play a part in 'thickening' the digits. The training accuracy and loss progression for the best performing $A2B$ ISTN is shown in figure 5.4 on page 165.

In the transfer from thin-*unslanted* to thin-*slanted* domains, it is expected that the necessary transformation is an affine shearing of the digit. The appearance of the digit, its thickness, should remain unchanged. Thus, the STN is expected to play an important role in this transformation. The best

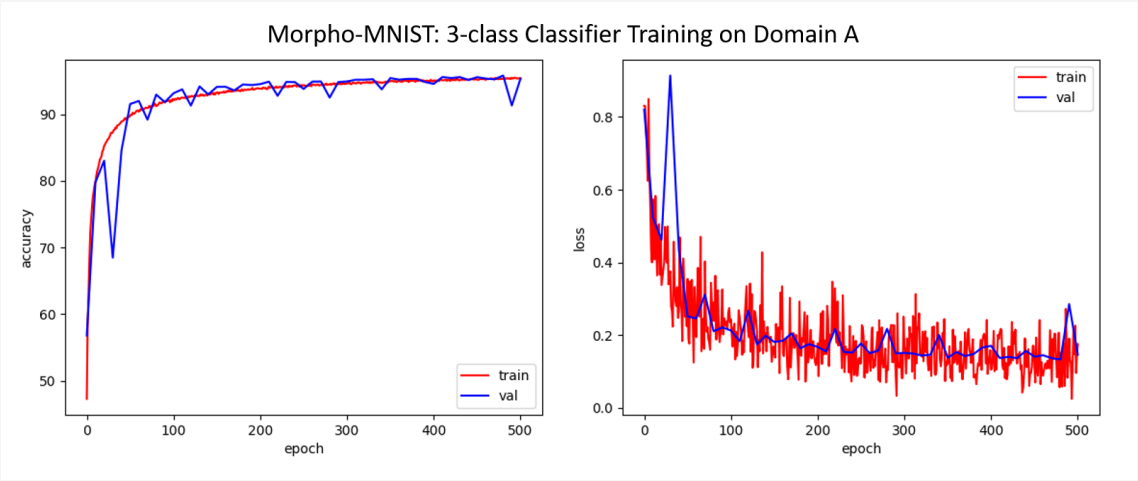


Figure 5.3: 3-class Classifier Training Progression on Morpho-MNIST data. Accuracy (left) and loss (right) progression over training epochs for 3-class 'disease' classifier trained on Morpho-MNIST domain A (thin-unslanted digits).

Table 5.11: Baseline accuracies of Morpho-MNIST 3-class classifier 3-class 'disease' classifier trained on domain (A) thin-unslanted - and evaluated on the remaining domains (B) thick-unslanted, (C) thin-slanted, and (D) thick-slanted.

	Domain Accuracy (%))			
Classifier	**A**	**B**	**C**	**D**
$\mathcal{T}_A$	**95.8**	41.2	45.7	32.8

recovery for this shift occurs when training the classifier on outputs from the B-spline ISTN combinations $\mathcal{T}_{A2C}(C) = 93.4\,\%$. However, the STNs alone are able to transform the image sufficiently to recover the performance of the classifier to similar levels. Thus the spatial transformations are most important for transfer from domain $\mathbf{A}$ to $\mathbf{C}$. Note that the expected affine STN transformation is practically equal with the B-spline STNs when the STNs are considered alone and not in combination with the ITN. It is inferred that the image transformation is trying to compete with the spatial transformation when the ISTN is employed, but only spatial transforms are required.

The final domain transformation takes *thin-unslanted* to *thick-slanted* digits. There are elements of both intensity and spatial changes needed to successfully transform between these domains. It is expected that the ISTN will provide the best results in this case. However, despite the ISTN combinations giving rise to good recovery of the classifier, it is seen that the ITN alone is able to produce the greatest change to the task model performance. The classifier accuracy, when finetuned from $\mathcal{T}_A$, improves from $32.8\,\%$ to $83.1\,\%$ on the affine ISTN output and up to $84.6\,\%$ when using the ITN alone.

The ITN is able to perform the transformation using intensity changes alone: creating high inten-

Table 5.12: DA results for 3-class classifier on Morpho-MNIST. Images transferred from classifier domain A: 'thin-unslanted' to three target domains. Accuracies (%) shown for classifiers $\mathcal{T}_{A2B}(B)$, $\mathcal{T}_{A2C}(C)$ and $\mathcal{T}_{A2D}(D)$. Each is retrained on the ISTN output from scratch ($\mathrm{Acc_s}$) and finetuned ($\mathrm{Acc_f}$) from $\mathcal{T}_A$. Δ is the model improvement from baselines. Control-point spacings indicated in brackets for B-Spline STNs. First row is the original classifier $\mathcal{T}_A$ performance without DA. $\mathcal{T}_A(A) = 95.8\,\%$

Target		(B) **Thick** Unslanted				(C) Thin **Slanted**				**(D) Thick Slanted**			
ITN	**STN**	**$\mathrm{Acc_s}$**	**Δ**	**$\mathrm{Acc_f}$**	**Δ**	**$\mathrm{Acc_s}$**	**Δ**	**$\mathrm{Acc_f}$**	**Δ**	**$\mathrm{Acc_s}$**	**Δ**	**$\mathrm{Acc_f}$**	**Δ**
✗	✗	41.2				45.7				32.8			
✓	✗	**79.0**	**37.8**	**83.3**	**42.1**	83.4	37.7	83.3	37.6	**82.4**	**49.6**	**84.6**	**51.8**
✗	**Affine**	52.4	11.2	68.9	27.7	92.4	46.7	93.0	47.3	54.8	22.0	64.8	32.0
✗	**B-spline (4)**	39.0	-2.2	54.4	13.2	92.1	46.4	93.1	47.4	36.0	3.2	57.2	24.4
✗	**B-spline (8)**	49.2	8.0	61.5	20.3	92.5	46.8	92.3	46.6	37.0	4.2	61.8	29.0
✓	**Affine**	78.8	37.6	77.1	35.9	86.7	41.0	88.4	42.7	81.9	49.1	83.1	50.3
✓	**B-spline (4)**	66.3	25.1	75.8	34.6	**92.7**	**47.0**	91.0	45.3	79.3	46.5	82.7	49.9
✓	**B-spline (8)**	69.5	28.3	77.2	36.0	91.8	46.1	**93.4**	**47.7**	79.0	46.2	80.8	48.0

sities where needed, and effectively erasing the digit to background intensity where appropriate. It is suspected that the ITN in this experiment is powerful enough to encompass a range of complex brightness and contrast transformations such that it is able to perform well where a simple spatial transform would suffice. More complex image structure, such as that in brain MRI may provide more of a challenge for the ITN. The Morpho-MNIST experiment may be too simplistic, to demonstrate this DS recovery in a decomposed manner, *i.e.* intensity and spatial transformations. It is also possible that the B-Spline STN with fixed CP-spacing is not necessarily the optimal architecture to use in these experiments.

Examples of ISTN training progression for $\mathcal{I}_{A2B}$, $\mathcal{I}_{A2C}$ and $\mathcal{I}_{A2D}$ are displayed in figure 5.5 on page 166, figure 5.6 on page 166 and figure 5.7 on page 167 respectively. The progression shows that in the transfer from *thin*-unslanted to *thick*-unslanted domains, the ISTN $\mathcal{I}_{A2B}$ very quickly learns the transformations it must apply to successfully move between domains. There is evidence that the transformation respects the class features (*i.e.* fractures and swellings) though some fractures may be eliminated by the thickening of the digit on either side. This may cause a small number of False Negative (FN) predictions when evaluating the 3-class classifier. Increasing the width of the fracture, currently at 1 pixel, may help to alleviate these false negative predictions. Swellings seem to be well transformed as they are enlarged by the same proportion as the digit itself.

Outputs during $\mathcal{I}_{A2C}$ (thin-*unslanted* domain to thin-*slanted* domain) training show that the the ISTN finds the necessary shearing transformation, but perhaps not so quickly as $\mathcal{I}_{A2B}$ finds the thickening transformation. It is observed that some digits are continually perturbed to find the

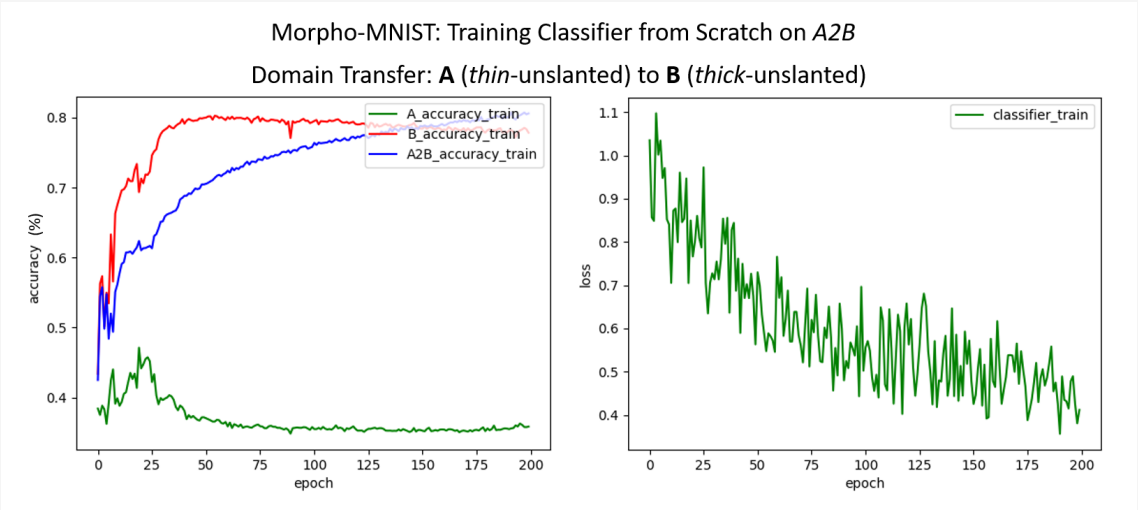


Figure 5.4: 3-class Classifier Training Progression: Thin-unslanted to Thick-unslanted Domains. Accuracy (left) and loss (right) progression over training epochs for 3-class 'disease' classifier trained on ISTN transformed output $A2B$ (*thin*-unslanted to *thick*-unslanted domains). Results shown for ITN with no STN.

correct slant, whilst others settle much more quickly. In the *thin-unslanted* to *thick-slanted* ISTN $\mathcal{I}_{A2D}$ training, a similar patter emerges: the transformation for the thickness of the digit is very quickly discovered whilst the shear of the digit may take a little longer to learn.

There is little difference seen between the B-spline STNs with CP-spacings of 4 and 8 pixels. These values correspond to $1/8$ and $1/4$ of the 32×32 image respectively. Where there is a noticeable difference, the spacing of 8 pixels more often provides slightly improved classifier performance. It may be expected that having CPs spaced too close together will generate artifacts that cause unwanted and unrealistic perturbations of the image. Additionally, it is likely to take longer to train the ISTN to regress displacements for a greater number of CPs. Perhaps training for a greater number of epochs would flatten out the discrepancies.

The loss curves for ISTN and discriminator training on the domain transfer $\mathbf{A}$ to $\mathbf{B}$ is shown in figure 5.8 on page 167. The ISTNs and discriminators learn well in the first epoch of training resulting in a quick descent towards some minimum of the objective functions. However, the 'minimax' game [58] played between the generative and discriminator models causes the loss for both to fluctuate rather than smoothly converge to a minimum. Extremes are seen where some batches of input data cause larger errors in the output which are backpropagated through the models. This fluctuation causes difficulty in deciding when the ISTN is properly trained. We cannot use the loss on the ISTN to decide when to stop training. To overcome this, the method in this work trains a task model ($\mathcal{T}_{S2T}$) in parallel and evaluates it on the ISTN output $S2T$ at each validation step. The ISTN is trained such that its output $S2T$ gives the best performance on $\mathcal{T}_{S2T}(S2T)$.

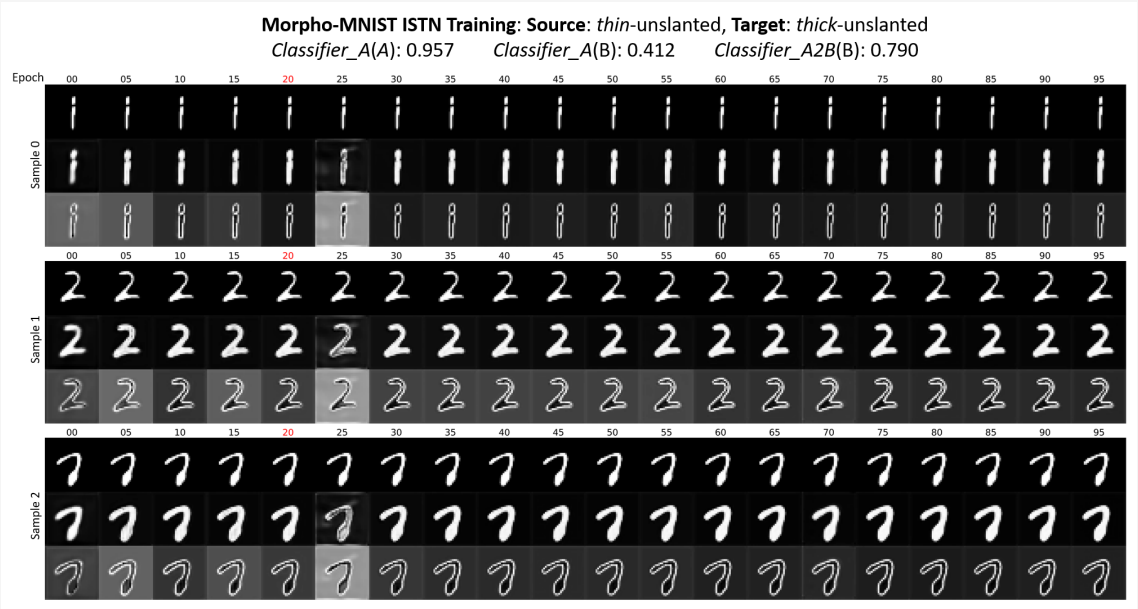


Figure 5.5: Morpho-MNIST ISTN Training Samples: thin-unslanted to thick-unslanted. Training progression for ISTN trained to transfer digits from domain A (thin-unslanted) to domain B (thick-unslanted). Training epochs progress from left-to-right. Each panel shows (top) a training sample; (middle) the ISTN output; and (bottom) the difference image. The epoch number is red is that at which the task model (classifier) achieves greatest performance on the domain B.

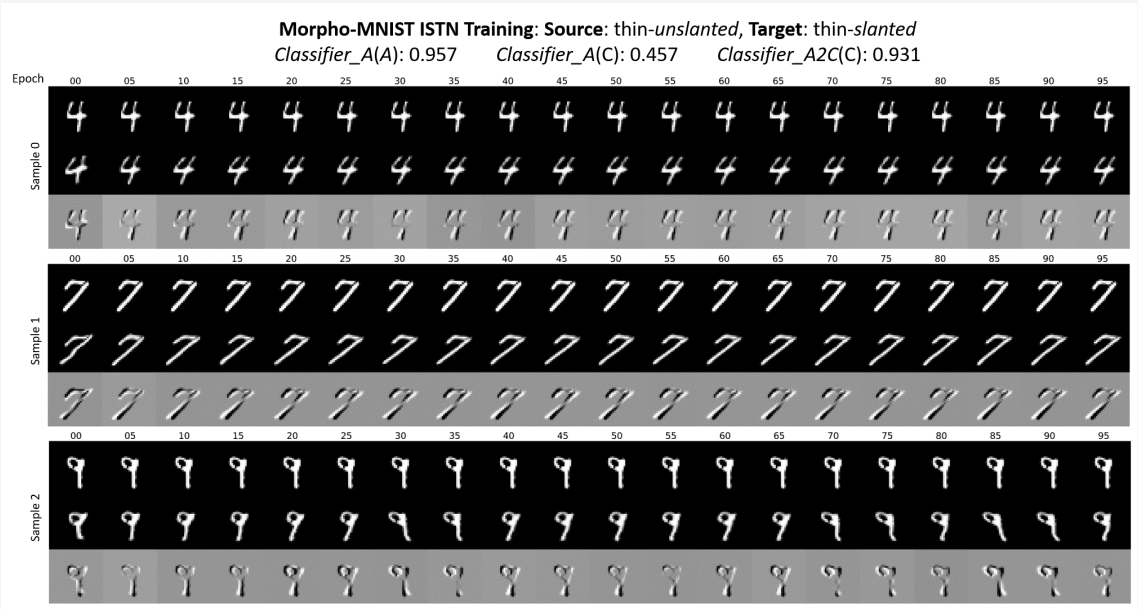


Figure 5.6: Morpho-MNIST ISTN Training Samples: thin-unslanted to thin-slanted. Training progression for ISTN trained to transfer digits from domain A (thin-unslanted) to domain C (thin-slanted). Training epochs progress from left-to-right. Each panel shows (top) a training sample; (middle) the ISTN output; and (bottom) the difference image.

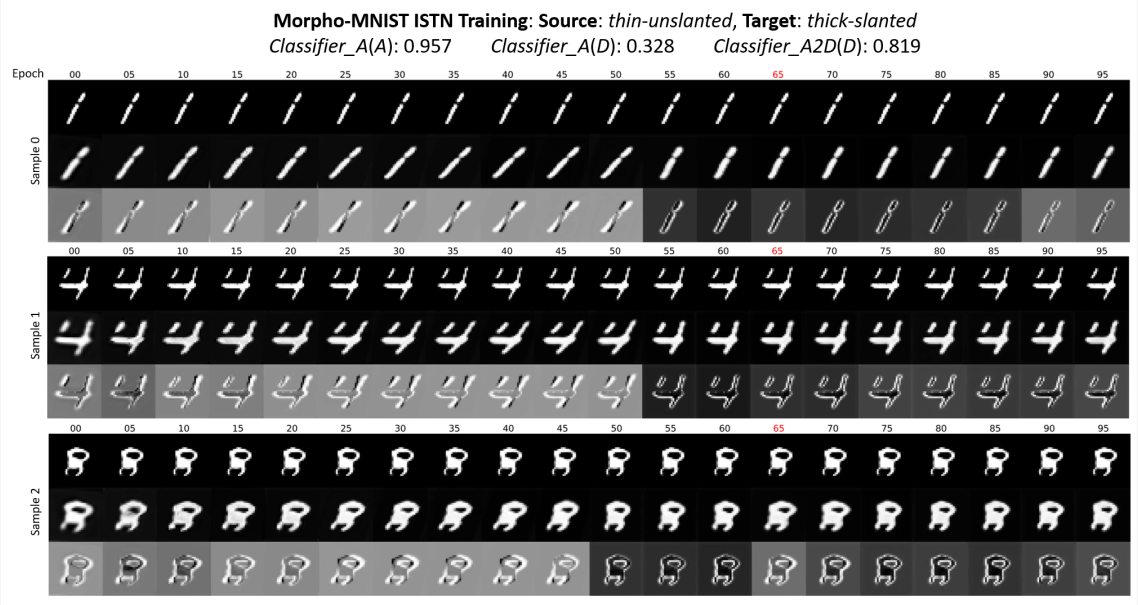


Figure 5.7: Morpho-MNIST ISTN Training Samples: thin-unslanted to thick-slanted. Training progression for ISTN trained to transfer digits from domain A (thin-unslanted) to domain D (thick-slanted). Training epochs progress from left-to-right. Each panel shows (top) a training sample; (middle) the ISTN output; and (bottom) the difference image. The epoch number is red is that at which the task model (classifier) achieves greatest performance on the domain D.

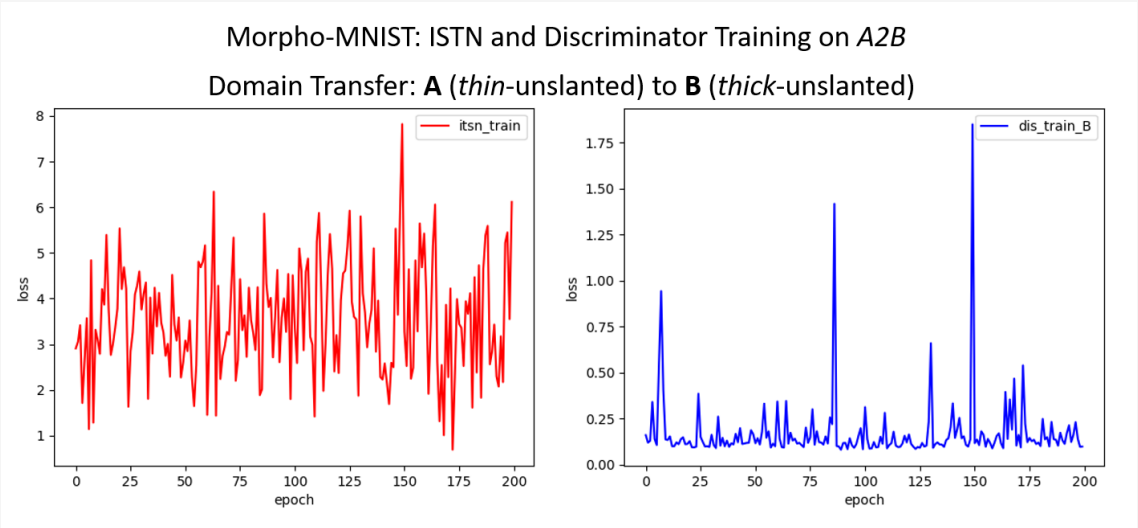


Figure 5.8: ISTN Training Progression: Thin-unslanted to Thick-unslanted Domains. (left) ISTN and (right) discriminator losses for ISTN training. Transfer between domain (A) *thin*-unslanted and (B) *thick*-unslanted. Results shown for ITN with no STN.

5.5.3 Brain MRI: Sex Classification Results.

Table 5.13: Baseline accuracies of Brain MRI sex classifiers Models trained on images from the UKBB $\mathcal{T}_U^{\text{sex}}$ and Cam-CAN $\mathcal{T}_C^{\text{sex}}$ and then evaluated on their respective training domains and opposing domains.

	Domain Accuracy (%)	
Regressor	UKBB (**U**)	Cam-CAN (**C**)
$\mathcal{T}_U^{\text{sex}}$	**84.3**	54.8
$\mathcal{T}_C^{\text{sex}}$	64.3	**91.6**

Two sex classifiers, outlined in section 5.4.5.2 on page 158, were trained separately on the UKBB and Cam-CAN datasets. Each classifier was trained for 100 epochs with a learning rate 0.0001. Loss and accuracy progression over the training period on the UKBB domain $\mathbf{U}$ and Cam-CAN domain $\mathbf{C}$ are shown in figure 5.9 on the following page. The task model $\mathcal{T}_U^{\text{sex}}$ trained on the UKBB dataset reaches a maximum classification accuracy $\mathcal{T}_U^{\text{sex}}(U) = 84.3\,\%$. The task model $\mathcal{T}_C^{\text{sex}}$ trained on the Cam-CAN dataset reaches a maximum performance $\mathcal{T}_C^{\text{sex}}(C) = 91.6\,\%$.

When $\mathcal{T}_U^{\text{sex}}$ and $\mathcal{T}_C^{\text{sex}}$ are tested on their opposing domains, a drop in model performance is observed with $\mathcal{T}_U^{\text{sex}}(C) = 54.8\%$ and $\mathcal{T}_C^{\text{sex}}(U) = 64.3\%$. These results are summarized in table 5.13. The drop in performance is likely due both to acquisition shift between scanners and population shift between age ranges.

5.5.4 Brain MRI: Age Regression Results.

Table 5.14: Baseline performance of brain MRI age regressors. Mean Absolute Errors (MAEs) of brain MRI age regressors trained on images from the UKBB $\mathcal{T}_U^{\text{reg}}$ and Cam-CAN $\mathcal{T}_C^{\text{reg}}$ evaluated on their respective training domains and opposing domains.

	Domain MAE (*years*)	
Regressor	UKBB (**U**)	Cam-CAN (**C**)
$\mathcal{T}_U^{\text{reg}}$	**0.041**	0.049
$\mathcal{T}_C^{\text{reg}}$	0.048	**0.041**

Two age regressors, outlined in section 5.4.5.3 on page 158, were trained separately on the UKBB and Cam-CAN datasets. Each regressor was trained for 100 epochs with a learning rate 0.0002. The Mean Squared Error (MSE) loss between the regressor-predicted age and the sample's Ground Truth (GT) is minimized by the Adam optimizer.

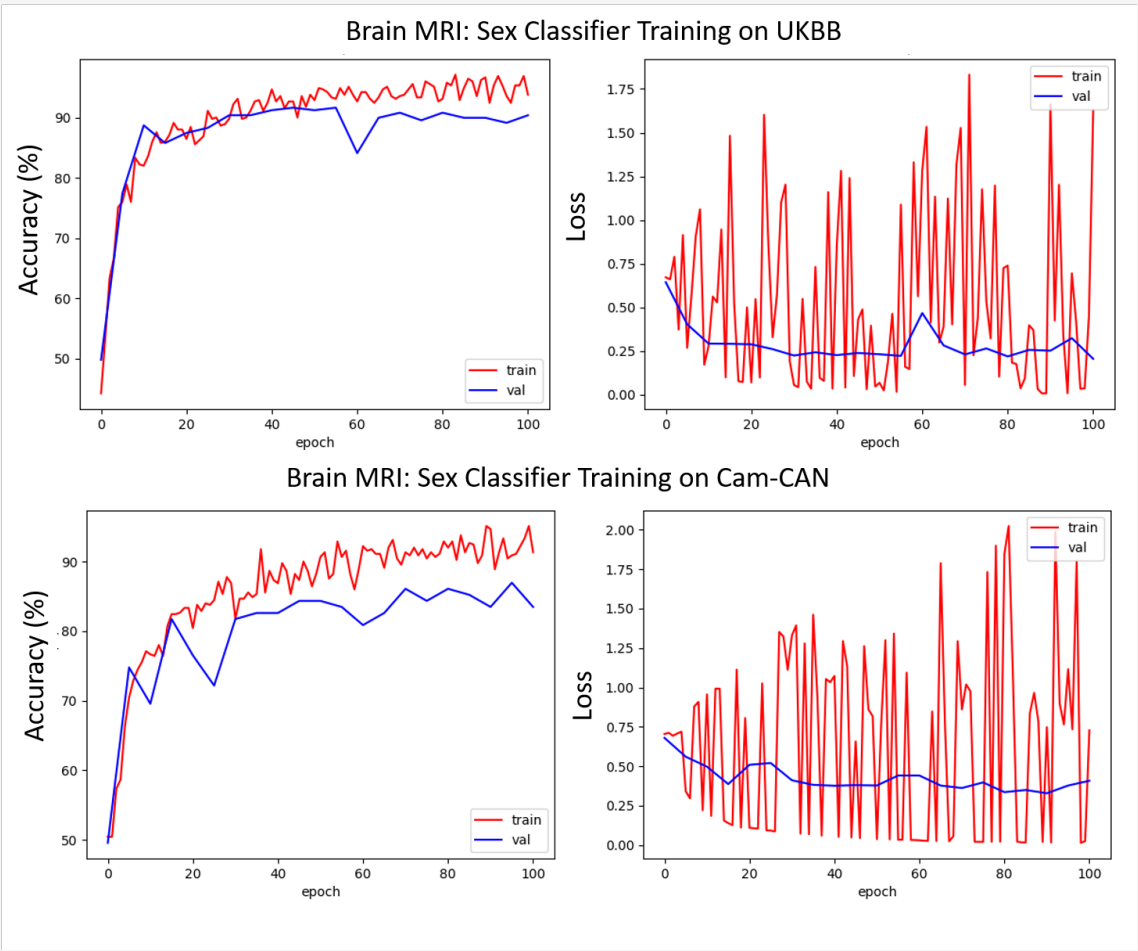


Figure 5.9: Sex Classifier Training Progression on Brain MRI. Accuracy (left) and loss (right) progression over training epochs for sex classifier trained on brain MRI from (top) the UKBB and (bottom) Cam-CAN.

To assess the performance of the regressor, two metrics are used. First the MAE loss is considered, where a lower MAE between the predicted and real ages is better. Second, a series of thresholds are applied to the absolute difference between predicted and real age for each example. Accuracy is defined by the number of examples whose prediction is within $t \in T = \{1, 2, 5\}$ years of the GT where t is the applied threshold from the set of thresholds T.

MSE loss and accuracy progression over the training period on the UKBB domain $\mathbf{U}$ and Cam-CAN domain $\mathbf{C}$ are shown in figure 5.10 on the next page. The task model $\mathcal{T}_U^{\text{reg}}$ trained on the UKBB dataset reaches a best MAE of $\mathcal{T}_U^{\text{age}}(U) = 4.1$ years. The task model $\mathcal{T}_C^{\text{reg}}$ trained on the Cam-CAN dataset also reaches a best MAE of $\mathcal{T}_C^{\text{reg}}(C) = 4.1$ years.

When $\mathcal{T}_U^{\text{reg}}$ and $\mathcal{T}_C^{\text{reg}}$ are tested on their opposing domains, a drop in model performance is observed with $\mathcal{T}_U^{\text{reg}}(C) = 4.9$ years and $\mathcal{T}_C^{\text{reg}}(U) = 4.8$ years. These results are summarized in table 5.14 on the preceding page. The drop in performance is likely due to acquisition shift between scanners as the samples are all in the same age range and have the same sex balance.

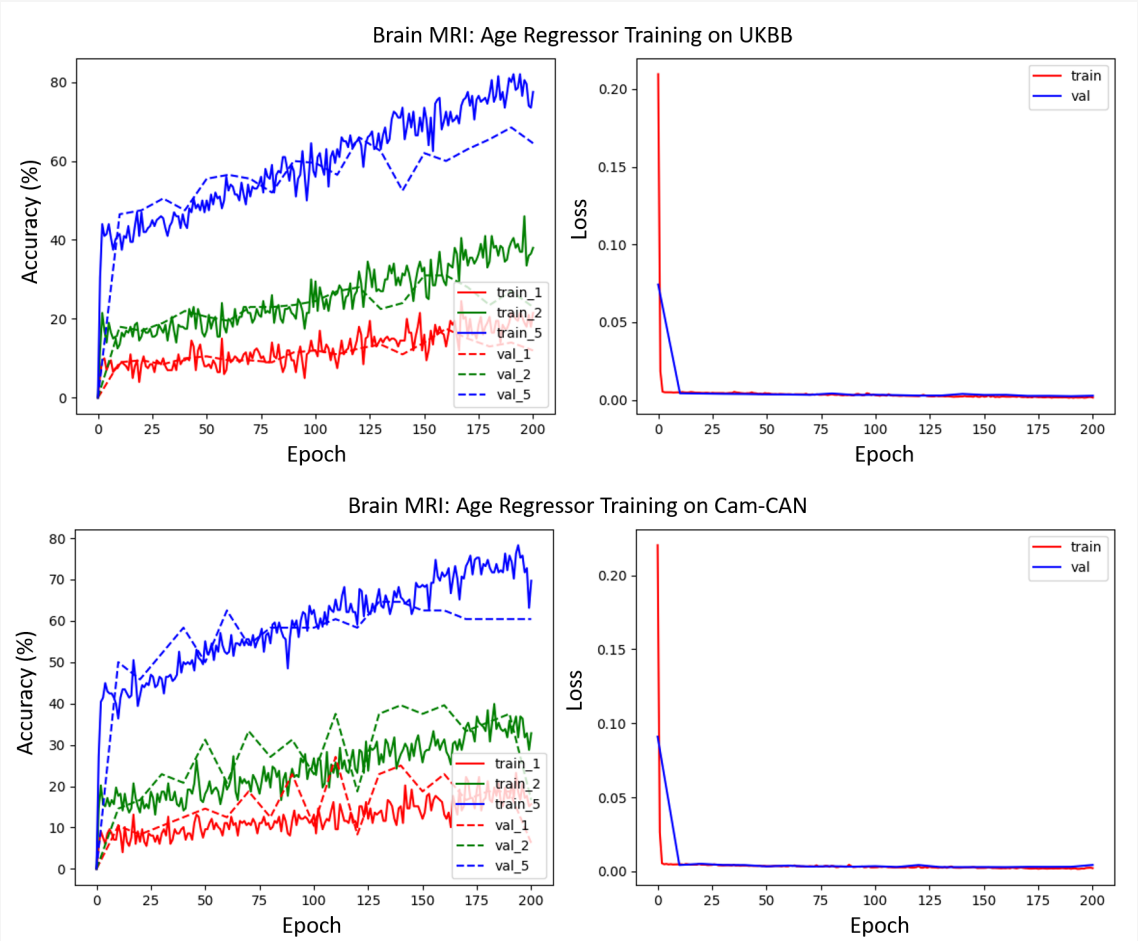


Figure 5.10: Age Regressor Training Progression on Brain MRI. Accuracy (left) and loss (right) progression over training epochs for age regressor trained on brain MRI from (top) the UKBB and (bottom) Cam-CAN. Accuracy is calculated using a threshold on the Mean Absolute Error (MAE) between regressor prediction and GT. Thresholds are 1, 2 and 5 years as shown.

Table 5.15: DA results for sex classifier on brain MRI. Sex classification results on 3D Brain MRI using ISTN trained in parallel with sex classification task. Accuracies (%) shown for classifiers $\mathcal{T}^{\text{sex}}_{U2C}(C)$ from the UKBB source and $\mathcal{T}^{\text{sex}}_{C2U}(U)$ from the Cam-CAN source. Each is retrained on the ISTN output from scratch (Acc_s) and finetuned (Acc_f) from $\mathcal{T}^{\text{sex}}_{U}$ and $\mathcal{T}^{\text{sex}}_{C}$ respectively. Δ is the model improvement from baselines. Control-point spacings are indicated in brackets for B-Spline STNs. First row is the performance of the original classifiers $\mathcal{T}^{\text{sex}}_{U}(C)$ and $\mathcal{T}^{\text{sex}}_{C}(U)$ without DA. Performance of $\mathcal{T}^{\text{sex}}_{U}(U) = 84.3\,\%$ and $\mathcal{T}^{\text{sex}}_{C}(C) = 91.6\,\%$.

Source		**UKBB**								**Cam-CAN**							
Target		**Cam-CAN**								**UKBB**							
Method		**Uni-ISTN**				**Bi-ISTN**				**Uni-ISTN**				**Bi-ISTN**			
ITN	**STN**	**Acc_s**	Δ	**Acc_f**	Δ	**Acc_s**	Δ	**Acc_f**	Δ	**Acc_s**	Δ	**Acc_f**	Δ	**Acc_s**	Δ	**Acc_f**	Δ
✗	✗	54.8				54.8				64.3				64.3			
✓	✗	79.1	24.3	72.2	17.4	**80.0**	**25.2**	80.8	26.0	**86.2**	21.9	78.2	13.9	80.8	16.5	79.9	15.6
✓	**A**	**80.9**	**26.1**	75.7	20.9	70.4	15.6	**82.4**	**27.6**	79.9	15.6	79.1	14.8	**82.4**	**18.1**	72.0	7.7
✓	**B(8)**	78.3	23.5	76.5	21.7	79.1	24.3	78.7	23.9	80.3	16.0	**84.5**	**20.2**	78.7	14.4	**80.8**	**16.5**
✓	**B(16)**	80.0	25.2	**78.3**	**23.5**	73.0	18.2	67.8	13.0	85.4	21.1	84.1	19.8	67.8	3.5	68.6	4.3

5.5.5 Brain MRI: DA with ISTN Results.

The ISTN combinations indicated in table 5.1 on page 148 were trained using both the unidirectional and bidirectional approaches outlined in section 5.4.3 on page 148 and shown in figure 5.1 on page 151.

In the unidirectional case, experiments were performed independently to train an ISTN $\mathcal{I}_{U2C}$ to transfer images from the UKBB to the Cam-CAN domain and an ISTN $\mathcal{I}_{C2U}$ to transfer images from the Cam-CAN to the UKBB domain.

For each of the classification and regression tasks, the task model $\mathcal{T}^{\text{task}}_{U2C}$ was trained in parallel with $\mathcal{I}_{U2C}$, both from scratch and finetuned from $\mathcal{T}^{\text{task}}_{U}$. Labels from the source domain $\mathbf{U}$ were used to evaluate $\mathcal{T}^{\text{task}}_{U2C}(C)$ and assess the recovery of the model from $\mathcal{T}^{\text{task}}_{U}(C)$. Similarly, $\mathcal{T}^{\text{task}}_{C2U}$ and $\mathcal{I}_{C2U}$ were trained in parallel, both from scratch and finetuned from $\mathcal{T}^{\text{task}}_{C}$. Labels from the source domain $\mathbf{C}$ were used to evaluate $\mathcal{T}^{\text{task}}_{C2U}(U)$ and assess the recovery of the model from $\mathcal{T}^{\text{task}}_{C}(U)$. Task model training in all cases was independent of the ISTN training. The ISTN was considered 'trained' when the task model trained on its output, $\mathcal{T}^{\text{task}}_{U2C}$ or $\mathcal{T}^{\text{task}}_{C2U}$, reached maximum performance.

5.5.5.1 Brain MRI: DA applied to Sex Classification.

Table 5.15 shows quantitative results for each ISTN combination with unidirectional and bidirectional training methods. In the UKBB to Cam-CAN direction the sex classifier regains 21-26% of performance when trained on ISTN output with the affine ISTN performing best. The bidirectional

CycleGAN training approach does not contribute to substantially improved performance of the ISTN as measured by the task model. As with the Morpho-MNIST experiments, the smaller spacing of control-points in the B-Spline STN yields slightly reduced performance in the unidirectional training scheme, but when trained bidirectionally, the larger spacing does not perform so well. Fine-tuning the sex classifier does not show any consistently substantial benefit over training one from scratch on the ISTN output.

Setting the Cam-CAN dataset as the source domain, it is seen that the recovery in performance is smaller. However, the original classifier $\mathcal{T}_C^{sex}$ performs extremely well on its own domain and recovery to this level may be much more difficult. Though the gains are smaller, the ISTNs are able to recover performance to a slightly higher accuracy than in the UKBB to Cam-CAN direction due to the higher baseline. The bidirectional training scheme is also not shown to be beneficial in this case.

It is expected that the bidirectional training should yield ISTNs that work together to perform DA in both directions. The data in table 5.15 on the previous page demonstrate that regardless of which site is set as the source domain, the corresponding ISTNs trained with the bidirectional training approach appear to train classifiers that come towards a common performance level.

Qualitative examples for the four ISTN combinations are shown in figure 5.11 on the following page. It is seen that the B-spline ISTNs produce transformed images which allow the retrained sex classifier to reach greater predictive performance. There is little difference between the B-spline ISTNs with CP-spacings of 8 and 16 voxels. This may imply that both spacings converge to a similar deformation. It is observed that both B-Spline ISTNs are applying transformations at a local level.

In comparison, the output from the affine ISTN appears to perform only intensity changes at a global tissue-level causing a difference in tissue contrast between the input and transformed images. This is consistent with the result when applying only the ITN without an STN.

The transformations that can be performed by an affine STN are limited to global translations, rotation, scaling and shearing of the image. All images used in this work are already affinely registered which removes much of the global spatial variation between the domains. Therefore it is understood that the ITN alone and the affine ISTN should return similar transformed image outputs as demonstrated.

To gain more insight into the changes being made by the components of the ISTN, the Structural Similarity Index (SSIM) [170] between images from the source domain and the images transformed by the ISTN is assessed. SSIM measures how local means (luminance) and local variances (contrast) of the images compare along with the normalized standard deviations (structure). The product of these metrics gives the overall SSIM. The three components are calculated within a window with shape $[1, 8, 8, 8]$ for each image in the source domain. The SSIM is measured both after the

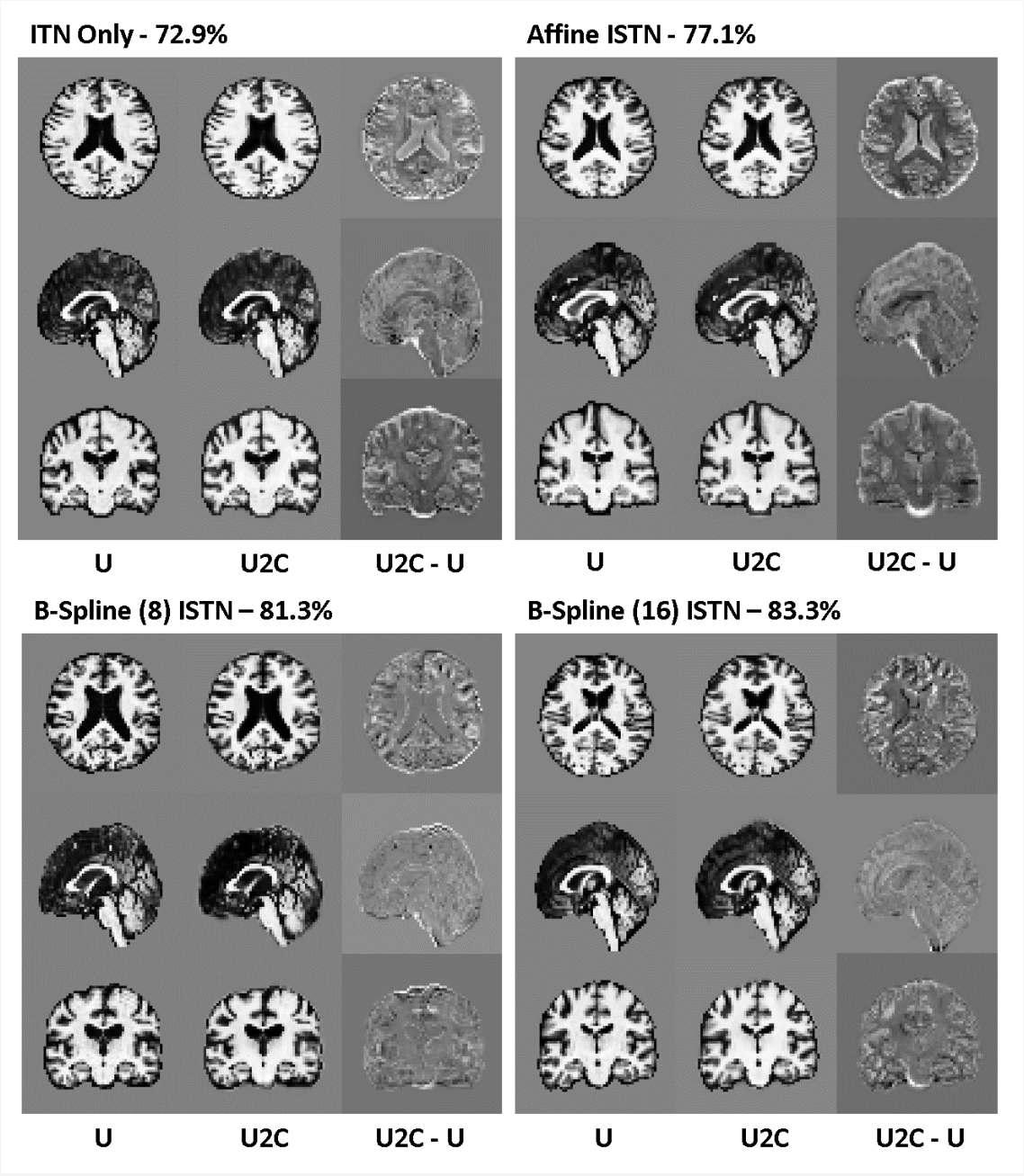


Figure 5.11: Sample Output for Each ISTN Combination Trained on Brain MRI. (top-left to bottom-right) itn only, affine, B-Spline (Control Point (CP) spacing of 8 voxels) and B-Spline (CP spacing of 16 voxels). Each example shows (left-to-right) the source domain image from UKBB, the image transformed to the target domain (Cam-CAN), and a difference image between the two. Shown for (top-to-bottom) axial, sagittal and coronal views.

image is passed through the ITN and after the STN. The metrics at each point are averaged for the dataset. It is seen that, on the whole, the changes made in the UKBB to Cam-CAN direction are greater than those being made in the opposite direction. The ITN without an STN seems affect greater changes to the image than when combined the affine STN. This is seen in some results where the ITN is capable of recovering performance even in the absence of a STN. It appears as

Table 5.16: Structural Similarity Index before, during and after DA in brain MRI Structural Similarity Index (SSIM) and its components, Luminance (L), Contrast (C) and Structure (S), calculated after source domain images pass through the ITN (*After ITN*) and the STN (*After STN*). Comparison is always against the original source domain image. Shown for both transformation directions: UKBB to Cam-CAN and Cam-CAN to UKBB. Results are an average over all images from the dataset after passing through the corresponding ISTN. Values in range $[0.0,\ 1.0]$ where 1.0 means no change at all from the original image. ITN with B-Spline STN at CP-spacing of 8 shows the largest changes in in SSIM corresponding to the greatest modification of appearance and structure.

	Direction	**UKBB to Cam-CAN**				**Cam-CAN to UKBB**			
Metric	**Stage**	**STN**				**STN**			
		None	**Affine**	**B(8)**	**B(16)**	**None**	**Affine**	**B(8)**	**B(16)**
Luminance (L)	**After ITN**	0.980	0.986	0.954	0.951	0.999	0.999	0.999	1.000
	Δ (%)	2.04	1.39	4.78	5.13	0.08	0.08	0.11	0.05
	After STN	0.980	0.986	0.945	0.948	0.999	0.998	0.999	0.999
	Δ (%)	2.04	1.44	5.76	5.44	0.08	0.15	0.10	0.07
Contrast (C)	**After ITN**	0.995	0.997	0.985	0.992	0.998	0.978	0.989	0.994
	Δ (%)	0.51	0.32	1.52	0.82	0.21	2.21	1.16	0.56
	After STN	0.995	0.995	0.986	0.989	0.998	0.977	0.994	0.996
	Δ (%)	0.51	0.52	1.42	1.08	0.21	2.34	0.57	0.36
Structure (S)	**After ITN**	0.996	0.996	0.984	0.989	0.996	0.991	0.989	0.995
	Δ (%)	0.42	0.38	1.60	1.13	0.41	0.89	1.10	0.53
	After STN	0.996	0.995	0.963	0.985	0.996	0.985	0.978	0.993
	Δ (%)	0.42	0.48	3.81	1.47	0.41	1.49	2.20	0.65
SSIM ($LxCxS$)	**After ITN**	0.971	0.980	0.931	0.936	0.993	0.970	0.977	0.989
	Δ (%)	2.94	2.03	7.40	6.78	0.70	3.14	2.35	1.13
	After STN	0.971	0.977	0.908	0.929	0.993	0.962	0.972	0.989
	Δ (%)	2.94	2.37	9.84	7.59	0.70	3.93	2.84	1.08

though the ITN does not perform luminance changes in order to transform an image from the Cam-CAN to UKBB domain, but focuses on contrast changes. This agrees with qualitative inspection of the images, that UKBB images offer slightly higher contrast than those from Cam-CAN. The STNs clearly, and correctly, have a greater effect on the structural component of SSIM than on the luminance or contrast. Overall, the B-Spline STN with CP-spacing of 8 produces the greatest change in SSIM, compared with the other combinations.

5.5.5.2 Brain MRI: DA applied to Age Regression.

The regressors trained on the two domains for the age regression task achieve good performance in terms of MAE. In both domains, around 60% of the validation data is correctly predicted to within

Table 5.17: DA results for age regressor on brain MRI. Age regression results on 3D Brain MRI using the ISTNs trained in parallel $\mathcal{I}^{\text{reg}}$ and the ISTNs trained during the sex-classification task $\mathcal{I}^{\text{sex}}$. MAE (years) shown for regressors (top) $\mathcal{T}^{\text{reg}}_{U2C}(C)$ and (bottom) $\mathcal{T}^{\text{reg}}_{C2U}(U)$. Each was retrained on the ISTN output from scratch (MAE_s) and finetuned (MAE_f) from $\mathcal{T}^{\text{reg}}_{U}$ and $\mathcal{T}^{\text{reg}}_{C}$ respectively. Δ is the model improvement from baselines. Control-point spacings are indicated in brackets for B-Spline STNs. First row is the performance of the original regressors $\mathcal{T}^{\text{reg}}_{U}(C)$ and $\mathcal{T}^{\text{reg}}_{C}(U)$ without DA. Performance is equal for $\mathcal{T}^{\text{reg}}_{U}(U) = 0.041$ years and $\mathcal{T}^{\text{reg}}_{C}(C) = 0.041$ years.

Source		**UKBB**							
Target		**Cam-CAN**							
ISTN		$\mathcal{I}^{\text{reg}}_{U2C}$				$\mathcal{I}^{\text{sex}}_{U2C}$			
ITN	**STN**	**MAE$_\text{s}$**	Δ	**MAE$_\text{f}$**	Δ	**MAE$_\text{s}$**	Δ	**MAE$_\text{f}$**	Δ
✗	✗	0.0490				0.0490			
✓	✗	0.0538	-0.0048	0.0506	-0.0016	0.0762	-0.0272	0.0579	-0.0089
✓	Affine	0.0515	-0.0025	0.0560	-0.0070	0.0620	-0.0130	0.0567	-0.0077
✓	B-Spline(8)	**0.0483**	**0.0007**	**0.0477**	**0.0013**	**0.0480**	**0.0010**	**0.0454**	**0.0036**
✓	B-Spline(16)	0.0507	-0.0017	0.0481	0.0009	0.0513	-0.0023	0.0725	-0.0235

Source		**Cam-CAN**							
Target		**UKBB**							
ISTN		$\mathcal{I}^{\text{reg}}_{C2U}$				$\mathcal{I}^{\text{sex}}_{C2U}$			
ITN	**STN**	**MAE$_\text{s}$**	Δ	**MAE$_\text{f}$**	Δ	**MAE$_\text{s}$**	Δ	**MAE$_\text{f}$**	Δ
✗	✗	0.048				0.048			
✓	✗	0.0511	-0.0031	0.0468	0.0012	0.0502	-0.0022	0.0554	-0.0074
✓	Affine	0.0541	-0.0061	0.0481	-0.0001	0.0486	-0.0006	0.0660	-0.0180
✓	B-Spline(8)	0.0479	0.0001	**0.0466**	**0.0014**	**0.0475**	**0.0005**	**0.0486**	**-0.0006**
✓	B-Spline(16)	**0.0473**	**0.0007**	0.0490	-0.0010	0.0482	-0.0002	0.0518	-0.0038

a 5 year window either side of the GT. When evaluated on the opposing domains, the regressors still perform well with only a very subtle drop in performance.

Table 5.17 shows quantitative results for each ISTN combination with the unidirectional training scheme. Bidirectional training was not conducted as the results in the sex classification task were seen to offer no clear benefit over the unidirectional scheme.

The B-spline ISTN with the more compact CPs recoups the largest performance of 0.1 years on the MAE. Unlike the sex classification and the Morpho-MNIST experiments, the smaller spacing of CPs in the B-Spline STN yields slightly improved performance in the unidirectional training scheme. Finetuning the age regressor does not show any substantially consistent benefit over training one from scratch on the ISTN output. Due to the smaller loss in performance after DS, it is more difficult to recover the regressor performance than the classifier performance, but some recovery is still

possible.

The ISTNs were trained independently of the task model such that the ISTNs trained during the age regression task should be similar to those from the sex classification task. Thus, the ISTN is agnostic to the task model being trained. To investigate this, new regressors were trained using the output from the ISTNs trained in the sex classification task. Results are also shown in table 5.17 on the preceding page.

The ISTNs trained during the sex classification experiment show potential to be as good as, and better than, those trained in parallel with the regressor. The variation may be caused by random initialization differences. However, it is noted that the regression task models trained on sex-classification ISTN outputs have an overall larger error. This result may be expected due to the way in which the ISTN model is chosen during training. During the training process, the ISTN model and task model are evaluated at each validation step. The ISTN is saved only if the task model yields a better result than it has previously attained. Thus, the ISTN saved at the best sex classification result, even though the model has not converged, will be the model that generates the images that maximizes the classification result. This same model may not necessarily maximize the regression result. This is an issue which is difficult to overcome because deciding when to stop ISTN training is challenging without some input from the task model evaluation. The ISTN loss function is not straight-forward due to the adversarial training regime. This is clear from the ISTN training loss curves shown for the Morpho-MNIST experiment in figure 5.8 on page 167. A better method of evaluating ISTN convergence is needed.

In addition to the visualizing the training progression, as shown for the Morpho-MNIST experiments, the appearance and spatial transformations can be successively visualized. Figure 5.12 on the following page demonstrates the visualization of the appearance transformation applied after the ITN, followed by the visualization of the subsequent STN transformations with both Affine and B-Spline STNs. The difference images show the changes at each step. The deformation field applied by the STNs is visualized as a warped regular grid of points in order to demonstrate the spatial modifications made to the images. This level of explainability would not be possible using a domain-invariant feature learning approach.

The image in figure 5.12 on the next page is from the Cam-CAN dataset and the ISTN is transforming it into the UKBB domain. There appears to be some small scaling transformation being applied by the Affine STN implying some difference in brain size between the domains. A heterogeneous intensity change comes from the Affine STN transformation, which is larger than the changes applied by the ITN transformation. However, when the B-Spline STN is employed, the ITN delivers a more homogeneous change in intensity focused in the cortex and along boundaries. This may be indicative of the increase in contrast that was identified by studying the SSIM, and the qualitative verification that the UKBB images may exhibit higher contrast. This exploration of the transformations demonstrates that the choice of STN may be important.

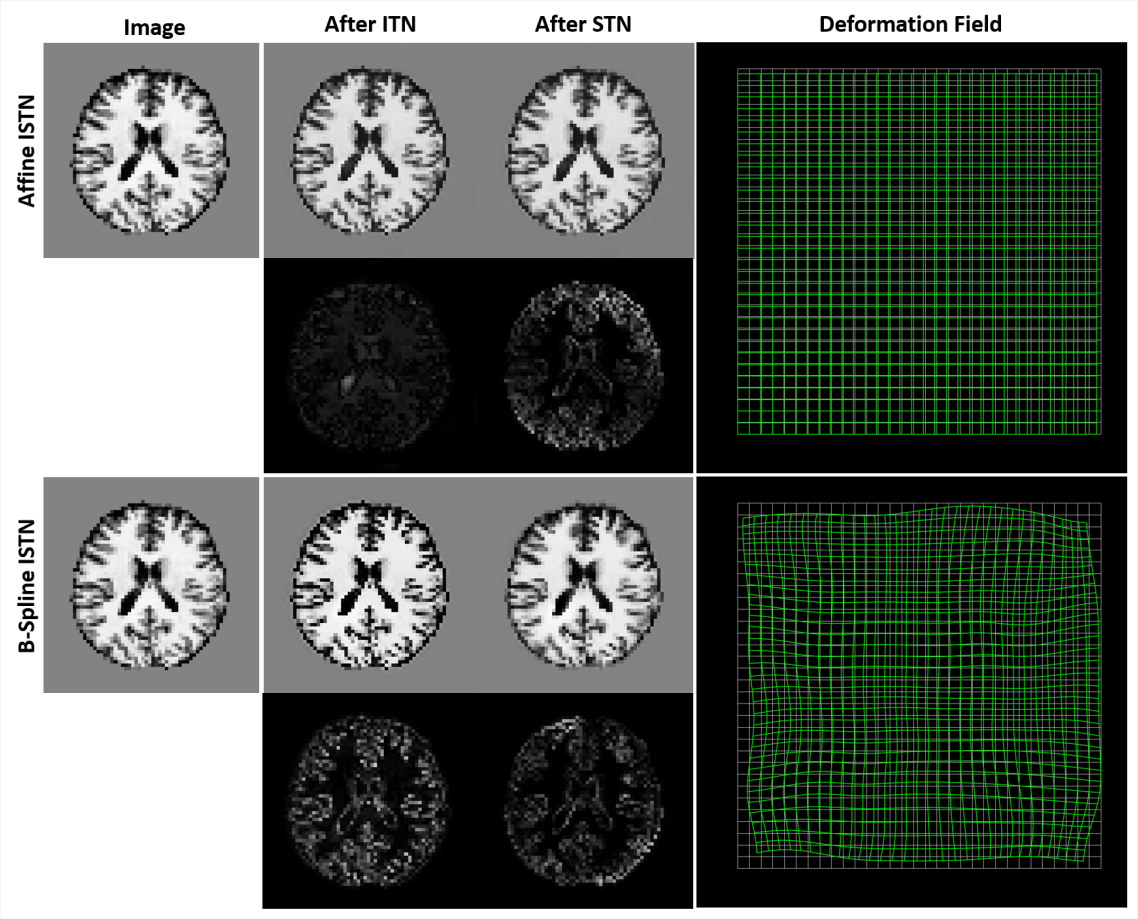


Figure 5.12: Example of Visual Explainability of ISTN. A sample of an affine (top) and B-Spline (bottom) ISTN transformation of the same image. Columns show (left) the original input image; the image, and difference image, after transformation by the ITN; the image, and difference image, after transformation by the STN; the deformation field.

5.6 Conclusions

Learning decomposed appearance and spatial transformations may be a viable approach to unsupervised, unpaired DA for the harmonization of multi-site medical imaging datasets. The ISTN framework employed in this work shows promising results in recovering model performance degradation due to population and acquisition shifts.

A ISTNs can be trained adversarially.

This work explored adversarially-trained ISTNs for model-agnostic DA. Training used feedback from a discriminator to continually improve the image and spatial transformation models. Results showed that the ISTNs trained in this manner were task agnostic allowing their output to be used to train different models. Thus, the mapping from source to target domain had been learned without being influenced by task-specific features. Adopting the CycleGAN-based bidirectional training scheme did not provide any substantial improvement in task model performance over the unidirectional approach.

Another direction to research could be to integrate the ISTN component in a fully end-to-end task-driven optimisation, where the ISTN and the task network are trained jointly. This approach would allow the ISTN to learn task-specific features and provide a focus for the domain harmonization.

B Model degradation due to DS is recoverable when ISTNs are used for DA.

Task models trained on ISTN outputs achieve a predictive performance which approaches single-site accuracies. This is demonstrated in results for both the sex classification and age regression tasks. Image-level DA seems suitable in cases of subtle domain shift caused by acquisition and population differences in multi-center studies.

C Image-level DA improves explainability over feature-level DA.

The learned image-level transformations help explainability, as the resulting images can be visually inspected and checked for plausibility: demonstrated in figure 5.11 on page 173 and figure 5.12 on the previous page. This is a distinct advantage over feature-level approaches. The image-level DA method has the advantage of modularity: once trained, the ISTN can be placed between the data loader and model training modules to adapt images on-the-fly. For feature-level DA techniques, task models are modified such that their latent features learn a joint-representation of the source and target domains. In the image-level DA scheme, task models do not need to undergo any alterations and their outputs can be directly compared with their source domain-trained counterparts.

A quantitative comparison to feature-level DA would be a natural next step for future work.

D A clearer measure of ISTN convergence is needed.

The ISTN trained in parallel with the sex classifier does not yield age regression performance on

par with training an ISTN in parallel with the age regressor itself. It was previously mentioned that this is likely the result of model checkpointing when the sex classifier is at its best performance. The problem here is the lack of a suitable method for understanding ISTN convergence. Indeed, this can be a problem with GAN training in general. The losses during ISTN training tend to drop quickly then fluctuate around some mean value. Therefore it is not possible to use the value of the loss to determine when the ISTN has finished training. In this work, the performance of the task model was used as a proxy for indicating ISTN convergence.

It would be useful to investigate this by training the ISTNs for much longer in order to determine their behaviour over a greater training period. Some other metrics for assessing ISTN convergence should also be investigated to retain the benefits of a model-agnostic training approach.

E Susceptibility to DS may be Task-Dependent.

Results for the age regression task on brain Magnetic Resonance (MR) show a small degradation in model performance compared with the sex classification task. The drop may go unnoticed if the regression model were deployed and evaluated on several different domains. However, results show that the change in classifier performance on different domains is likely to be stark. It is possible that susceptibility to DS may therefore be task-dependent.

Classification is achieved by training a model that learns some function or functions which discriminate between the available classes. There is a confidence or a margin of error associated with a model's class prediction which is related to the prediction's position from the decision boundary. A shift in the prediction due to a change in domain may be caused by the movement of examples across the decision boundary. The classifier is trained without taking into account the confidence of the classifier on each prediction. In the regression case, the learned function should follow the data closely such that outputs for new datapoints can be inferred. The regressor is trained with MSE loss acting as a confidence metric. A change in domain may shift predictions slightly, but on the whole the learned function is more generalized.

Chapter 6

Conclusions

This work has focused on the need for reliable Machine Learning (ML) in medical image analysis. Particular consideration has been given to the automated Quality Control (QC) of biomedical image segmentations at large scale and in the absence of Ground Truth (GT). An extension to the QC framework was also studied to enable real-time assessment of labelmaps in analysis workflows such as those in hospitals, pharmaceutical trials and population imaging initiatives. This work investigated an approach to mitigate the degradation in ML models performance due to Domain Shift (DS). Population and acquisition shifts were considered, and constrained transformations applied to reduce their effects.

The primary conclusions of this work are laid out here. Automated real-time QC procedures based on Reverse Classification Accuracy (RCA) can help to minimize the number of low quality automated segmentations that are sent for subsequent analysis, whilst performing Domain Adaptation (DA) using Image and Spatial Transformer Networks (ISTNs) is able to recover some degradation in model performance due to DS.

This chapter recounts the main conclusions of this work including its limitations. A discussion is begun on potential avenues for future work.

6.1 Main Conclusions.

This work has tackled the problem of reliable Machine Learning (ML) for medical imaging. It has focused on the automated Quality Control (QC) of Cardiac Magnetic Resonance Imaging (CMR) segmentations at large-scale, and in the absence of Ground Truth (GT), using a modified Reverse Classification Accuracy (RCA) approach. The possibility of automated real-time evaluation of segmentation quality has also been explored by employing Convolutional Neural Networks (CNNs). Domain Shift (DS) was investigated as a cause of model performance degradation, and Image and Spatial Transformer Networks (ISTNs) appraised as a method of Domain Adaptation (DA) in order to harmonize multi-site neuroimaging data. The primary conclusions are summarised here.

RCA Validation on Large-scale Imaging Data.

A RCA can predict the per-case quality of CMR segmentations generated by Random Forests (RFs) and CNNs.

B RCA can be preferable over rudimentary, manual QC scoring.

C The RCA run-time per test segmentation is too long for real-time applications.

D The size of the RCA reference set $\mathcal{R}$ is crucial.

E RCA does not give a one-to-one mapping between real and predicted evaluation metrics.

Real-time Quality Control Prediction with Convolutional Neural Networks.

A Predictions of quality for RF and CNN segmentations of CMR images can be achieved in real-time using CNN-based prediction models and RCA.

B The CNNs may learn features specific to the RF segmentor.

C The CNNs are currently not suitable for assessing fine-grain segmentation details.

D A CNN can assess segmentation quality without large labelled training sets.

E Methods cannot tell us about *why* a segmentation has failed.

Domain Adaptation with Image and Spatial Transformer Networks.

A Model degradation due to DS between brain Magnetic Resonance Imaging (MRI) datasets is mostly recoverable when ISTNs are used for DA.

B ISTNs can be trained adversarially.

C Image-level DA improves explainability over feature-level DA.

D A clearer measure of ISTN convergence is needed.

E Susceptibility to DS may be task-dependent.

6.1.1 Automated QC at Large Scale

Understanding when a segmentation has failed is important to ensure the validity of downstream analysis tasks such as biomarker extraction and quantification. In substantial imaging databases, such as those from population imaging studies, it is key to employ automated QC which can be done objectively, consistently, and accurately in the absence of GT.

This work has shown that the quality of automatically generated segmentations of CMR images can be evaluated on a per-case basis and at large scale. Experiments with datasets of 400, 4,800, and 7,250 RF segmentations, and 900 CNN segmentations demonstrated a 95-99% Whole-Heart (WH) binary classification accuracy on the Dice Similarity Coefficient (DSC) when a threshold of 0.70 was applied. Individual classes were shown to be well-classified except for the Left Ventricular Myocardium (LVM) which exhibits a more complex topology with two surfaces and small volume. Surface distance metrics show a low Mean Absolute Error (MAE) on WH, Left Ventricular Cavity (LVC), and Right Ventricular Cavity (RVC) segmentations.

The RCA methodology was modified in this work to perform 'multi-atlas' rather than 'single-atlas' registration. Rather than a creating a new labelmap from a majority vote of the registered reference set $\mathcal{R}$, a single atlas - the closest match - is chosen to act as a proxy for the predicted segmentation. This version of the RCA classifier has the advantage of only applying transformations to the atlases rather than modifying the test image and segmentation, which could introduce undesirable artifacts and registration errors.

The RCA process is slow, but the 7-10 minute run-time is acceptable for background processing in large databases. This work investigated the reference set $\mathcal{R}$ to determine if an atlas selection process could reduce the number of necessary registration and warping procedures whilst maintaining the excellent accuracy achieved in testing. A smaller number of atlases was found to maintain the high accuracy of the approach, but at the cost of a less varied set of examples in $\mathcal{R}$. Even with just 10 atlases, RCA achieved accuracies in the range $[0.87, 0.93]$ depending on the atlas selection. The method for choosing the atlases was not explored, however a procedure that may have merit could be to determine metrics such as Structural Similarity Index (SSIM) between the test image and atlas to compare their similarity and choose the best matches.

6.1.2 Real-time Automated QC

In the scheme where rapid analysis of segmentation results is desired, the current RCA approach is too slow. It may be important to provide immediate feedback about the quality of a generated segmentation such that some action may be taken by a user. It would be beneficial for a MRI technician to know when an acquired image leads to poor analysis results so that they may be able to acquire a new image whilst the subject is still within the scanner. This quick evaluation would save unnecessary stress for a patient who may need to be recalled for re-acquisition. It would save both scanner time and human resources, as well as limit the extra cost associated

with a re-scan.

A real-time QC framework would be advantageous in the pharmaceutical industry where high-throughput pipelines may be prevalent. Unsuccessful segmentations can be flagged for review by an expert or sent for re-segmentation with an alternative algorithm.

The output from an immediate automated QC assessment framework could be used as a signal during segmentor training. When the segmentation result from the segmentor model is assessed, in real-time, the QC module could provide feedback to the segmentor such that it generated better quality results the next time around.

The decision to use CNNs in this work was not only for their predictive power, but also their millisecond-runtime at inference. Once trained, these models can perform QC automatically $\sim$1000 times faster than with RCA.

This work showed that the DSC for RF segmentations can be directly predicted from the test image-segmentation pair with MAE=0.03 and standard deviation σ=0.04. With a threshold DSC=0.70, the binary classification accuracy of 'good' and 'poor' segmentations using direct predictions of DSC was 96.5%.

In the likely scenario where a large labelled dataset is not available on which to train a direct DSC prediction model, this work demonstrated a method to predict RCA scores. As RCA requires only a small set of manually labelled reference images, it is possible to generate a large enough training set of images with RCA predictions on which a quality prediction CNN can be trained. Such a model was demonstrated in section 3.4 and achieved a MAE=0.14 and standard deviation σ=0.09. At the threshold DSC=0.70, 91% accuracy is achieved classifying 'good' and 'poor' quality segmentations. The True Positive Rate (TPR)=0.879 was good and the False Positive Rate (FPR) negligible. The analysis showed that the model was equally good at predicting the RCA score for high and low quality segmentations.

RCA scores do not give a one-to-one mapping to GT DSCs, but this work has shown they are well-correlated and able to classify segmentations with good accuracy. Therefore, predicting RCA scores may be a viable alternative to direct DSC prediction in analysis pipelines that have a need for real-time results. Pharmaceutical trials can use the QC CNN to analyze results of automated segmentation algorithms which are triggered when new data is streamed into their analysis platforms. During acquisition, the predictions of RCA scores may be used to immediately inform the operator of the likely quality of subsequent analysis steps if the currently acquired image were to be used. In conjunction with other modules, a real-time assessment of segmentation quality could be used to inform the operator of the adjustments necessary to improve the downstream analysis task.

6.1.3 Reliability of ML Models.

The performance of a segmentor may be influenced by the distribution from which the test sample is drawn. If the test distribution is different to that on which the model was trained, the resulting segmentation may be of poor quality due to Domain Shift (DS). Therefore some method of forcing harmonization between training and test distributions is necessary in order to ensure the reliability of these ML models when deployed in new domains.

This work demonstrated DS between multi-site imaging data. It was seen that changes due to population shift may have had a greater influence than more subtle scanner effects. This may be explained by considering the structural and anatomical changes between cohorts of subjects, *e.g.* young and old, which may be much larger than the variation in contrast and appearance between different scanners. The age regression task showed a much smaller drop in performance due to DS where the data were age- and sex-matched. This is in comparison to the sex classification experiment where the age range was much larger for one of the datasets. The results of this work and [56] show that considerations of sex and age are important when constructing datasets for ML tasks in medical imaging.

ISTNs were shown to help recover some of the performance lost due to DS. It was seen that the Image Transformer Network (ITN) component was able to recover much of the performance when no Spatial Transformer Network (STN) was present, this suggests that scanner effects played some part in these particular datasets. The choice of STN may be important to obtain the best mapping between domains. The B-Spline STN was able to offer local geometric transformations of the image, whilst global transformations could be applied with an affine STN. This work presented results showing that image-level DA is able to perform Data Harmonization (DH) such that models trained on one site can be evaluated on another with a much smaller drop in performance.

The constrained Image-to-Image (I2I) translation approach to DA has the benefit of inherent visual explainability. In contrast to I2I translation approaches that broadly apply Deep Learning (DL) frameworks wholesale to the image, the method evaluated in this work, which employs ISTNs, constrains transformations to some predefined parameters. In scenarios where the transformations caused by DS are known, and can be reduced to appearance and spatial transformations, an ISTN may beneficial. Using an affine STN allows the global transformation parameters to be estimated. Local transformations can be investigated by visualizing the deformation field found with the B-Spline STN. Difference images can be taken before and after the ITN, and after the STN components in order to gain insight into the appearance and structural transformations being applied.

Classification is often performed by thresholding on some predicted continuous value. For example, at the final classification layer in a DL-based classifier, the floating-point values are passed through some activation function, *e.g.* `sigmoid` or `softmax`. Such functions force the values into a certain range *e.g.* $[0.0, 1.0]$ and a threshold is applied to the output. In the binary case, the

threshold may be numerical, assigning classes based on whether the output is greater or less than some value. The threshold may also be statistical, setting the class according to which output element has the maximum value. When predicting continuous values, a small change from the GT contributes to the MAE on the prediction. In classification tasks small perturbations in the output, before thresholding, may cause many samples to cross the decision boundary leading to an incorrect classification. It would therefore be expected that variations in the predictions due to DS may adversely affect the classification task, particularly in the multi-class case. The classification tasks in chapter 5 show larger discrepancies due to DS than the regression tasks, which may be partially due to this effect. It is difficult to confirm this without further investigation as the difference may only be due to the larger effect of population shift which is not present in the regression task.

6.2 Limitations.

Whilst this work has made some important contributions to the field of automated Quality Control (QC) and reliable Machine Learning (ML) for multi-site medical imaging models, it is not without limitations. The primary limitations are laid out here.

Reverse Classification Accuracy (RCA) Validation on Large-scale Imaging Data.

A Registration, along with segmentation, is an ill-posed problem which may introduce inaccuracies into the RCA process.

B Manual QC scores are based on only 3 Cardiac Magnetic Resonance Imaging (CMR) slices for a single class, and scores for each slice were not recorded.

Real-time Quality Control Prediction with Convolutional Neural Networks.

A Automated segmentations from only a single algorithm, *i.e.* Random Forests (RFs), are used to train the model.

B QC Convolutional Neural Network (CNN) serves only as a metric of segmentation quality, it must be combined with some feedback mechanism to be truly useful.

Domain Adaptation with Image and Spatial Transformer Networks.

A Analysis applies only to datasets of the same modality and anatomy.

B Further investigation of the specific appearance and spatial changes is needed.

C Only 2 large-scale datasets were considered in this work.

6.2.1 Reverse Classification Accuracy.

The process of RCA relies heavily on image registration. There is a registration between each atlas image in the reference set $\mathcal{R}$ and the test image followed by a warping of each atlas labelmap to the test image space. In section 2.4.1.5 on page 82 it was observed that performing image resampling after a transformation requires some degree of interpolation: linear for the image data and nearest-neighbour for the discrete labelmaps. This process may introduce errors in the labelmap which contribute to discrepancies in the calculated Dice Similarity Coefficient (DSC) between the warped labelmap and test segmentation. The registration process itself could be affected if the reference set is not representative of the distribution from which the test image is drawn.

The method of image registration will also have some bearing on the result. In this work, affine registration is performed between the images in $\mathcal{R}$ and the test image. However, applying rigid registration and reducing the degrees of freedom may degrade the image alignment and cause

an underestimation of the DSC. Conversely, performing non-linear registration may increase the processing time which is undesirable. A balance must be struck between accuracy and processing time which may be helped by carefully selecting the atlases that will be used in RCA. Atlas selection was not considered in this work, but a discussion on the size of $\mathcal{R}$ has been presented.

The modified RCA method was applied to CMR segmentations where manual QC was performed on only 3 slices per segmentation. The scores were given per-image rather than per-slice and only for the Left Ventricular Myocardium (LVM) class. The RCA process applied in this work showed difficulty in accurately scoring the individual LVM class, but excellent results in the Whole-Heart (WH) case. With access to manual QC of other structures or scores on a greater number of slices per image, the RCA predictions may exhibit greater correlation. It should be noted that RCA was able to extract poor quality segmentations missed by the manual rater.

6.2.2 Quality Control CNN.

The dataset used to train the QC CNN was constructed by consciously degrading the quality of a RF segmentor. Whilst the QC method was able to predict RCA scores on this dataset, it may lack generalizability to segmentations generated with other algorithms. The model trained on this dataset is not able to give fine-grained assessments of segmentation quality which would be necessary when applied to CNN segmentations where subtle errors are likely to be at boundaries between classes.

Some of the experiments in this work would benefit from having additional datasets on which to experiment. In the QC CNN, training on data from multiple sites would give the network added robustness and increased generalization to data from unseen sites.

6.2.3 Domain Adaptation.

The work on Domain Adaptation (DA) would also benefit from obtaining a third or more datasets such that combinations of sites can be investigated with regard to Domain Shift (DS). The UK Biobank (UKBB) and Cambridge Centre for Ageing and Neuroscience (Cam-CAN) datasets are acquired using similar protocols and parameters. Consideration of T1-weighted brain Magnetic Resonance Imaging (MRI) datasets acquired with stark differences in parameters, *e.g.* B_0, would give insight into the true ability of Image and Spatial Transformer Networks (ISTNs) to perform DA between sites. Applying DA to datasets containing different contrasts would also be a logical next step.

It may be the case that some larger dataset is composed of many smaller datasets which vary in scale. The variation and imbalance in such a dataset is likely to benefit from DA approaches in order to harmonize the images and allow task models to focus on extracting the most discriminative features for the task.

Further constraints on the Image Transformer Network (ITN) may help to further clarify the modifications made to the contrast and luminance of the image in terms of intensity ranges or non-linear corrections.

6.3 Further Work.

Whilst Reverse Classification Accuracy (RCA) is clearly a useful metric for assessing the quality of a segmentation, it seems that the ever-increasing use of Deep Learning (DL) and the desire for real-time analysis may force further developments in Convolutional Neural Network (CNN)-based Quality Control (QC). However, in the case where background processing is acceptable, RCA may still be explored. Further investigation is needed into atlas selection procedures in order to select the best reference set, which is required for RCA. Due to the challenges associated with image registration, other methods of finding the best-matching reference image should also be explored. Metrics like Structural Similarity Index (SSIM) and Mutual Information (MI) may be advantageous in these areas.

In the real-time case, training the QC CNN with segmentations generated by an ensemble of methods would be the next priority. This would allow the model to generalize to a variety of segmentation 'styles' where variations in quality may manifest differently for different methods.

The Domain Adaptation (DA) work needs to be verified with data from more domains. Primarily, more neuroimaging datasets would be acquired to reproduce the success of Image and Spatial Transformer Networks (ISTNs) on the given tasks. An investigation could also be made into the effectiveness of training an ISTN on many domains. Such a model would transfer new data to a common domain on which the task model had already been trained ISTNs.

Deploying the QC CNN in clinic or for pharmaceutical trials would be ambitious a this stage. Further testing needs to be done on the generalizability of the model and the specific segmentation packages utilized in individual scanners would need to be considered. Working with partners in local hospitals may help to understand the specific needs of Magnetic Resonance Imaging (MRI) operators and the role that QC CNN may play in improving the quality of image segmentations.

The reliability of Machine Learning (ML) models may depend heavily on the similarity between training and test distributions, but the ability to transform between domains will help to rectify performance degradation due to Domain Shift (DS).

Whilst the methods outlined in this work must be considered further, they provide a beginning to ensure that all stakeholders retain confidence in the models that are being used to inform clinical decision making.

References

[1] Adalsteinsson, D. and Sethian, J. A. "A Fast Level Set Method for Propagating Interfaces". In: *Journal of Computational Physics* 118.2 (May 1995), pp. 269–277. ISSN: 0021-9991. DOI: 10.1006/JCPH.1995.1098.

[2] Adamson, A. S. and Smith, A. "Machine learning and health care disparities in dermatology". In: *JAMA Dermatology* 154.11 (Nov. 2018), pp. 1247–1248. ISSN: 21686068. DOI: 10.1001/jamadermatol.2018.2348.

[3] Alaiz-Rodríguez, R. and Japkowicz, N. "Assessing the impact of changing environments on classifier performance". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5032 LNAI. Springer, Berlin, Heidelberg, 2008, pp. 13–24. ISBN: 3540688218. DOI: 10.1007/978-3-540-68825-9_2.

[4] Alfaro-Almagro, F., Jenkinson, M., Bangerter, N. K., et al. "Image processing and Quality Control for the first 10,000 brain imaging datasets from UK Biobank". In: *NeuroImage* 166 (Feb. 2018), pp. 400–424. ISSN: 10959572. DOI: 10.1016/j.neuroimage.2017.10.034.

[5] Aljabar, P., Heckemann, R. A., Hammers, A., et al. "Multi-atlas based segmentation of brain images: Atlas selection and its effect on accuracy". In: *NeuroImage* 46.3 (July 2009), pp. 726–738. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2009.02.018.

[6] Asman, A. J. and Landman, B. A. "Non-local statistical label fusion for multi-atlas segmentation". In: *Medical Image Analysis* 17.2 (Feb. 2013), pp. 194–208. ISSN: 1361-8415. DOI: 10.1016/J.MEDIA.2012.10.002.

[7] Babalola, K. O., Cootes, T. F., Twining, C. J., et al. "3D brain segmentation using active appearance models and local regressors". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5241 LNCS. PART 1. Springer, Berlin, Heidelberg, 2008, pp. 401–408. ISBN: 354085987X. DOI: 10.1007/978-3-540-85988-8_48.

[8] Bai, W., Shi, W., O'Regan, D. P., et al. "A probabilistic patch-based label fusion model for multi-atlas segmentation with registration refinement: application to cardiac MR images." In: *IEEE transactions on medical imaging* 32.7 (July 2013), pp. 1302–15. ISSN: 1558-254X. DOI: 10.1109/TMI.2013.2256922.

[9] Bai, W., Sinclair, M., Tarroni, G., et al. "Automated cardiovascular magnetic resonance image analysis with fully convolutional networks 08 Information and Computing Sciences 0801 Artificial Intelligence and Image Processing". In: *Journal of Cardiovascular Magnetic Resonance* 20.1 (Sept. 2018), p. 65. ISSN: 1532429X. DOI: 10.1186/s12968-018-0471-x.

[10] Bardera, A., Feixas, M., Boada, I., et al. "Registration-based segmentation using the information bottleneck method". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 4478 LNCS. PART 2. Springer, Berlin, Heidelberg, 2007, pp. 130–137. ISBN: 9783540728481. DOI: 10.1007/978-3-540-72849-8_17.

[11] Ben-Hur, A., Horn, D., Siegelmann, H., et al. "Support Vector Clustering". In: *Journal of Machine Learning Research* 2 (2001), pp. 125–137. DOI: 10.1162/15324430260185565.

[12] Bousmalis, K., Silberman, N., Dohan, D., et al. "Unsupervised pixel-level domain adaptation with generative adversarial networks". In: *Proceedings - 30th IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR 2017. Vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 95–104. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.18. arXiv: 1612.05424.

[13] Boykov, Y., Veksler, O., and Zabih, R. "Fast approximate energy minimization via graph cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (Nov. 2001), pp. 1222–1239. ISSN: 01628828. DOI: 10.1109/34.969114.

[14] Bradley, D. and Roth, G. "Adaptive Thresholding using the Integral Image". In: *Journal of Graphics Tools* 12.2 (Jan. 2007), pp. 13–21. ISSN: 1086-7651. DOI: 10.1080/2151237x.2007.10129236.

[15] Branco, P., Torgo, L., and Ribeiro, R. P. "A Survey of Predictive Modeling on Imbalanced Domains". In: *ACM Comput. Surv.* 49.2 (Aug. 2016). ISSN: 0360-0300. DOI: 10.1145/2907070.

[16] Brehler, M., Islam, A., Vogelsang, L. O., et al. "Coupled active shape models for automated segmentation and landmark localization in high-resolution CT of the foot and ankle". In: *Proceedings of SPIE–the International Society for Optical Engineering*. Vol. 10953. SPIE-Intl Soc Optical Eng, Mar. 2019, p. 23. ISBN: 9781510625532. DOI: 10.1117/12.2515022.

[17] Breiman, L. "Bagging predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140. ISSN: 08856125. DOI: 10.1007/bf00058655.

[18] Breiman, L. "Random forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/A:1010933404324.

[19] Brown, L. G. "A Survey of Image Registration Techniques". In: *ACM Comput. Surv.* 24.4 (Dec. 1992), pp. 325–376. ISSN: 0360-0300. DOI: 10.1145/146370.146374.

[20] Cai, Y., Sharma, H., Chatelain, P., et al. "SonoEyeNet: Standardized fetal ultrasound plane detection informed by eye tracking". In: *Proceedings - International Symposium on Biomedical Imaging*. Vol. 2018-April. IEEE Computer Society, May 2018, pp. 1475–1478. DOI: 10.1109/ISBI.2018.8363851.

[21] Carapella, V., Jiménez-Ruiz, E., Lukaschuk, E., et al. "Towards the Semantic Enrichment of Free-Text Annotation of Image Quality Assessment for UK Biobank Cardiac Cine MRI Scans". In: *MICCAI Workshop on Large-scale Annotation of Biomedical data and Expert Label Synthesis (LABELS)*. Springer International Publishing, 2016, pp. 238–248. DOI: 10.1007/978-3-319-46976-8_25.

[22] Cardinal, M.-H. R., Soulez, G., Tardif, J.-C., et al. "Fast-marching segmentation of three-dimensional intravascular ultrasound images: A pre- and post-intervention study". In: *Medical Physics* 37.7Part1 (June 2010), pp. 3633–3647. ISSN: 00942405. DOI: 10.1118/1.3438476.

[23] Castro, D. C., Walker, I., and Glocker, B. *Causality matters in medical imaging*. Dec. 2019. arXiv: 1912.08142.

[24] Castro, D. C., Tan, J., Kainz, B., et al. "Morpho-MNIST: Quantitative Assessment and Diagnostics for Representation Learning". In: *Journal of Machine Learning Research* 20.178 (2019), pp. 1–29. DOI: http://hdl.handle.net/10044/1/63396.

[25] Cavallaro, A., Gelasca, E. D., and Ebrahimi, T. "Objective evaluation of segmentation quality using spatio-temporal context". In: *IEEE International Conference on Image Processing*. Vol. 3. 2002. DOI: 10.1109/icip.2002.1038965.

[26] Cheng, R., Turkbey, B., Gandler, W., et al. "Atlas based AAM and SVM model for fully automatic MRI prostate segmentation". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*. Vol. 2014. Institute of Electrical and Electronics Engineers Inc., Nov. 2014, pp. 2881–2885. ISBN: 9781424479290. DOI: 10.1109/EMBC.2014.6944225.

[27] Chow, L. S. and Paramesran, R. *Review of medical image quality assessment*. May 2016. DOI: 10.1016/j.bspc.2016.02.006.

[28] Cieslak, D. A. and Chawla, N. V. "A framework for monitoring classifiers' performance: when and why failure occurs?" In: *Knowledge and Information Systems* 18.1 (Jan. 2009), pp. 83–108. ISSN: 0219-1377. DOI: 10.1007/s10115-008-0139-1.

[29] Collins, D. L., Neelin, P., Peters, T. M., et al. "Automatic 3d intersubject registration of mr volumetric data in standardized talairach space". In: *Journal of Computer Assisted Tomography* 18.2 (1994), pp. 192–205. ISSN: 15323145. DOI: 10.1097/00004728-199403000-00005.

[30] Cootes, T. F., Edwards, G. J., and Taylor, C. J. "Active appearance models". In: *Computer Vision — ECCV'98*. Ed. by Burkhardt, H. and Neumann, B. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 484–498. ISBN: 978-3-540-69235-5.

[31] Cootes, T. F., Cooper, D. H., Taylor, C. J., et al. "Trainable method of parametric shape description". In: *Image and Vision Computing* 10.5 (June 1992), pp. 289–294. ISSN: 02628856. DOI: 10.1016/0262-8856(92)90044-4.

[32] Cootes, T., Edwards, G., and Taylor, C. J. "Comparing Active Shape Models with Active Appearance Models". In: *Procedings of the British Machine Vision Conference 1999*. Vol. 1. British Machine Vision Association, 1999, pp. 18.1–18.10. ISBN: 1-901725-09-X. DOI: 10.5244/C.13.18.

[33] Cootes, T., Taylor, C., Cooper, D., et al. "Active Shape Models-Their Training and Application". In: *Computer Vision and Image Understanding* 61.1 (Jan. 1994), pp. 38–59. ISSN: 10773142. DOI: 10.1006/cviu.1995.1004.

[34] Correia, P. and Pereira, F. "Objective evaluation of relative segmentation quality". In: *IEEE International Conference on Image Processing*. Vol. 1. 2000, pp. 308–311. DOI: 10.1109/icip.2000.900956.

[35] Cortes, C. and Vapnik, V. "Support-vector networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1007/bf00994018.

[36] Coupé, P., Manjón, J. V., Fonov, V., et al. "Patch-based segmentation using expert priors: Application to hippocampus and ventricle segmentation". In: *NeuroImage* 54.2 (Jan. 2011), pp. 940–954. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2010.09.018.

[37] Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. USA: Wiley-Interscience, 1991, p. 18. ISBN: 0471062596.

[38] Crimi, A., Bakas, S., Kuijf, H., et al., eds. *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Vol. 11383. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-11722-1. DOI: 10.1007/978-3-030-11723-8.

[39] Criminisi, A. and Shotton, J. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer London, 2013. ISBN: 9781447149293. DOI: https://doi.org/10.1007/978-1-4471-4929-3.

[40] Dai, L., Li, T., Shu, H., et al. "Automatic brain tumor segmentation with domain adaptation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11384 LNCS. Springer Verlag, Sept. 2019, pp. 380–392. ISBN: 9783030117252. DOI: 10.1007/978-3-030-11726-9_34.

[41] Demidenko, E. "Kolmogorov-smirnov test for image comparison". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3046 LNCS.PART 4 (2004), pp. 933–939. ISSN: 03029743. DOI: 10.1007/978-3-540-24768-5_100.

[42] Deng, J., Dong, W., Socher, R., et al. "ImageNet: A large-scale hierarchical image database". In: Institute of Electrical and Electronics Engineers (IEEE), Mar. 2010, pp. 248–255. DOI: 10.1109/cvpr.2009.5206848.

[43] Duffy, I. R., Boyle, A. J., and Vasdev, N. "Improving PET Imaging Acquisition and Analysis With Machine Learning: A Narrative Review With Focus on Alzheimer's Disease and On-

cology." In: *Molecular imaging* 18 (Jan. 2019), p. 1536012119869070. ISSN: 1536-0121. DOI: 10.1177/1536012119869070.

[44] Epstein, A. M., Stern, R. S., Tognetti, J., et al. "The Association of Patients' Socioeconomic Characteristics with the Length of Hospital Stay and Hospital Charges within Diagnosis-Related Groups". In: *New England Journal of Medicine* 318.24 (June 1988), pp. 1579–1585. ISSN: 15334406. DOI: 10.1056/NEJM198806163182405.

[45] Fabijańska, A., Strzecha, K., and Sankowski, D. "Determination of image segmentation quality". In: *The Experience of Designing and Application of CAD Systems in Microelectronics - Proceedings of the 9th International Conference, CADSM 2007*. 2007, pp. 439–441. ISBN: 9789665535874. DOI: 10.1109/CADSM.2007.4297611.

[46] Fan, J., Zeng, G., Body, M., et al. "Seeded region growing: An extensive and comparative study". In: *Pattern Recognition Letters* 26.8 (June 2005), pp. 1139–1156. ISSN: 01678655. DOI: 10.1016/j.patrec.2004.10.010.

[47] Fan, W. and Davidson, I. "Reverse Testing: An Efficient Framework to Select Amongst Classifiers under Sample Selection Bias". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*. New York, New York, USA: ACM Press, 2006, p. 147. ISBN: 1595933395. DOI: 10.1145/1150402.1150422.

[48] Fawcett, T. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (June 2006), pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010.

[49] Feng, Y., Zhao, H., Li, X., et al. "A multi-scale 3D Otsu thresholding algorithm for medical image segmentation". In: *Digital Signal Processing: A Review Journal* 60 (Jan. 2017), pp. 186–199. ISSN: 10512004. DOI: 10.1016/j.dsp.2016.08.003.

[50] Freedman, D., Pisani, R., and Purves, R. *Statistics: Fourth International Student Edition*. International student edition. W.W. Norton & Company, 2007. ISBN: 9780393930436.

[51] Freund, Y. and Schapire, R. E. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (Aug. 1997), pp. 119–139. ISSN: 00220000. DOI: 10.1006/jcss.1997.1504.

[52] Fry, A., Littlejohns, T. J., Sudlow, C., et al. "Comparison of Sociodemographic and Health-Related Characteristics of UK Biobank Participants with Those of the General Population". In: *American Journal of Epidemiology* 186.9 (Nov. 2017), pp. 1026–1034. ISSN: 14766256. DOI: 10.1093/aje/kwx246.

[53] Gatys, L. A., Ecker, A. S., and Bethge, M. "Image Style Transfer Using Convolutional Neural Networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-Decem. IEEE Computer Society, Dec. 2016, pp. 2414–2423. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.265.

[54] Gianfrancesco, M. A., Tamang, S., Yazdany, J., et al. *Potential Biases in Machine Learning Algorithms Using Electronic Health Record Data*. Nov. 2018. DOI: 10.1001/jamainternmed.2018.3763.

[55] Girshick, R., Donahue, J., Darrell, T., et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Sept. 2014, pp. 580–587. ISBN: 9781479951178. DOI: 10.1109/CVPR.2014.81. arXiv: 1311.2524.

[56] Glocker, B., Robinson, R., Castro, D. C., et al. "Machine Learning with Multi-Site Imaging Data: An Empirical Study on the Impact of Scanner Effects". In: (Oct. 2019). arXiv: 1910.04597.

[57] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016, p. 111.

[58] Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Ghahramani, Z., Welling, M., Cortes, C., et al. Curran Associates, Inc., 2014, pp. 2672–2680.

[59] Gordon, D., Farhadi, A., and Fox, D. "Re 3 : Real-time recurrent regression networks for visual tracking of generic objects". In: *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), pp. 788–795. ISSN: 23773766. DOI: 10.1109/LRA.2018.2792152.

[60] Goshtasby, A., Gage, S. H., and Bartholic, J. F. "A Two-Stage Cross Correlation Approach to Template Matching". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.3 (1984), pp. 374–378. ISSN: 01628828. DOI: 10.1109/TPAMI.1984.4767532.

[61] Goshtasby, A. "Piecewise linear mapping functions for image registration". In: *Pattern Recognition* 19.6 (Jan. 1986), pp. 459–466. ISSN: 00313203. DOI: 10.1016/0031-3203(86)90044-0.

[62] Goshtasby, A. "Image registration by local approximation methods". In: *Image and Vision Computing* 6.4 (Nov. 1988), pp. 255–261. ISSN: 02628856. DOI: 10.1016/0262-8856(88)90016-9.

[63] Greig, D. M., Porteous, B. T., and Seheult, A. H. "Exact Maximum A Posteriori Estimation for Binary Images". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 51.2 (Jan. 1989), pp. 271–279. ISSN: 00359246. DOI: 10.1111/j.2517-6161.1989.tb01764.x.

[64] Guera, D. and Delp, E. J. "Deepfake Video Detection Using Recurrent Neural Networks". In: *Proceedings of AVSS 2018 - 2018 15th IEEE International Conference on Advanced Video and Signal-Based Surveillance*. Institute of Electrical and Electronics Engineers Inc., Feb. 2019. ISBN: 9781538692943. DOI: 10.1109/AVSS.2018.8639163.

[65] Gulshan, V., Peng, L., Coram, M., et al. "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs". In: *JAMA - Journal of the American Medical Association* 316.22 (Dec. 2016), pp. 2402–2410. ISSN: 15383598. DOI: 10.1001/jama.2016.17216.

[66] Guo, C., Niu, K., Luo, Y., et al. "Intra-scanner and inter-scanner reproducibility of automatic white matter hyperintensities quantification". In: *Frontiers in Neuroscience* 13.JUL (2019). ISSN: 1662453X. DOI: 10.3389/fnins.2019.00679.

[67] Guo, L., Liu, X., Wu, Y., et al. "Research on the segmentation of MRI image based on multi-classification support vector machine". In: *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*. 2007, pp. 6019–6022. ISBN: 1424407885. DOI: 10.1109/IEMBS.2007.4353720.

[68] Hasan, S. M. and Ahmad, M. *Two-step verification of brain tumor segmentation using watershed matching algorithm*. Dec. 2018. DOI: 10.1186/s40708-018-0086-x.

[69] Hathaway, R. J. and Bezdek, J. C. "Fuzzy c-means clustering of incomplete data". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 31.5 (Oct. 2001), pp. 735–744. ISSN: 10834419. DOI: 10.1109/3477.956035.

[70] He, K., Zhang, X., Ren, S., et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-Decem. IEEE Computer Society, Dec. 2016, pp. 770–778. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[71] Heckemann, R. A., Hammers, A., Aljabar, P., et al. "The mirror method of assessing segmentation quality in atlas label propagation". In: *Proceedings - 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, ISBI 2009*. 2009, pp. 1194–1197. ISBN: 9781424439324. DOI: 10.1109/ISBI.2009.5193272.

[72] Hinton, G. E., Srivastava, N., Krizhevsky, A., et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0 (2012).

[73] Ho, T. K. "Random decision forests". In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. Vol. 1. IEEE Computer Society, 1995, pp. 278–282. ISBN: 0818671289. DOI: 10.1109/ICDAR.1995.598994.

[74] Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[75] Hoebel, K., Andrearczyk, V., Beers, A. L., et al. "An exploration of uncertainty information for segmentation quality assessment". In: *Medical Imaging 2020: Image Processing*. Ed. by Landman, B. A. and Išgum, I. Vol. 11313. International Society for Optics and Photonics. SPIE, Mar. 2020, p. 55. ISBN: 9781510633933. DOI: 10.1117/12.2548722.

[76] Hoffman, J., Tzeng, E., Park, T., et al. "{C}y{CADA}: Cycle-Consistent Adversarial Domain Adaptation". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Dy, J. and Krause, A. Vol. 80. Proceedings of Machine Learning Research. Stockholm, Sweden: PMLR, 2018, pp. 1989–1998.

[77] Hripcsak, G., Duke, J. D., Shah, N. H., et al. "Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers". In: *Studies in Health Technology and Informatics*. Vol. 216. IOS Press, 2015, pp. 574–578. ISBN: 9781614995630. DOI: 10.3233/978-1-61499-564-7-574.

[78] Hu, M. K. "Visual Pattern Recognition by Moment Invariants". In: *IRE Transactions on Information Theory* 8.2 (1962), pp. 179–187. ISSN: 21682712. DOI: 10.1109/TIT.1962.1057692.

[79] Huang, C., Wu, Q., and Meng, F. "QualityNet: Segmentation quality evaluation with deep convolutional networks". In: *VCIP 2016 - 30th Anniversary of Visual Communication and Image Processing*. Institute of Electrical and Electronics Engineers Inc., Jan. 2017. ISBN: 9781509053162. DOI: 10.1109/VCIP.2016.7805585.

[80] Ioffe, S. and Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. JMLR.org, 2015, pp. 448–456.

[81] Isola, P., Zhu, J. Y., Zhou, T., et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., Nov. 2017, pp. 5967–5976. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.632. arXiv: 1611.07004.

[82] Jaderberg, M., Simonyan, K., Zisserman, A., et al. "Spatial Transformer Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2017–2025.

[83] Jia, X., Jin, Y., Su, X., et al. "Domain-invariant representation learning using an unsupervised domain adversarial adaptation deep neural network". In: *Neurocomputing* 355 (Aug. 2019), pp. 209–220. ISSN: 18728286. DOI: 10.1016/j.neucom.2019.04.033.

[84] Jiang, J., Zheng, S., Toga, A. W., et al. "Learning based coarse-to-fine image registration". In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2008. ISBN: 9781424422432. DOI: 10.1109/CVPR.2008.4587396.

[85] Jonsson, B. A., Bjornsdottir, G., Thorgeirsson, T. E., et al. "Brain age prediction using deep learning uncovers associated sequence variants". In: *Nature Communications* 10.1 (Dec. 2019), pp. 1–10. ISSN: 20411723. DOI: 10.1038/s41467-019-13163-9.

[86] Jovicich, J., Czanner, S., Greve, D., et al. "Reliability in multi-site structural MRI studies: Effects of gradient non-linearity correction on phantom and human data". In: *NeuroImage* 30.2 (Apr. 2006), pp. 436–443. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2005.09.046.

[87] Justice, R. K., Stokely, E. M., Strobel, J. S., et al. "Medical image segmentation using 3D seeded region growing". In: *Medical Imaging 1997: Image Processing*. Ed. by Hanson, K. M. Vol. 3034. SPIE, Apr. 1997, pp. 900–910. DOI: 10.1117/12.274179.

[88] Kalet, A. M., Luk, S. M. H., and Phillips, M. H. "Quality assurance tasks and tools: The many roles of machine learning". In: *Medical Physics* (Mar. 2019). ISSN: 00942405. DOI: 10.1002/mp.13445.

[89] Kamnitsas, K., Baumgartner, C., Ledig, C., et al. "Unsupervised domain adaptation in brain lesion segmentation with adversarial networks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10265 LNCS. Springer Verlag, 2017, pp. 597–609. ISBN: 9783319590493. DOI: 10.1007/978-3-319-59050-9_47. arXiv: 1612.08894.

[90] Kamnitsas, K., Ledig, C., Newcombe, V. F., et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation". In: *Medical Image Analysis* 36 (Feb. 2017), pp. 61–78. ISSN: 13618423. DOI: 10.1016/j.media.2016.10.004. arXiv: 1603.05959.

[91] Kang, H. C., Lee, J., and Shin, J. "Automatic four-chamber segmentation using level-set method and split energy function". In: *Healthcare Informatics Research* 22.4 (Oct. 2016), pp. 285–292. ISSN: 2093369X. DOI: 10.4258/hir.2016.22.4.285.

[92] Kass, M., Witkin, A., and Terzopoulos, D. "Snakes: Active contour models". In: *International Journal of Computer Vision* 1.4 (Jan. 1988), pp. 321–331. ISSN: 09205691. DOI: 10.1007/BF00133570.

[93] Kohlberger, T., Singh, V., Alvino, C., et al. "Evaluating Segmentation Error without Ground Truth". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*. 2012, pp. 528–536. DOI: 10.1007/978-3-642-33415-3_65.

[94] Kohli, M. D., Summers, R. M., and Geis, J. R. "Medical Image Data and Datasets in the Era of Machine Learning—Whitepaper from the 2016 C-MIMI Meeting Dataset Session". In: *Journal of Digital Imaging* 30.4 (Aug. 2017), pp. 392–399. ISSN: 1618727X. DOI: 10.1007/s10278-017-9976-3.

[95] Koutroumbas, K. and Theodoridis, S. *Pattern Recognition*. Elsevier, 2009, p. 203. DOI: 10.1016/B978-1-59749-272-0.X0001-2.

[96] Kuglin, C. D. "The Phase Correlation Image Alignment Methed". In: *Proc. Int. Conference Cybernetics Society* (1975), pp. 163–165.

[97] LeCun, Y., Boser, B. E., Denker, J. S., et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. Ed. by Touretzky, D. S. Morgan-Kaufmann, 1990, pp. 396–404.

[98] LeCun, Y., Bottou, L., Bengio, Y., et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219. DOI: 10.1109/5.726791.

[99] Lee, M. C., Oktay, O., Schuh, A., et al. "Image-and-Spatial Transformer Networks for Structure-Guided Image Registration". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11765 LNCS. Springer, Oct. 2019, pp. 337–345. ISBN: 9783030322441. DOI: 10.1007/978-3-030-32245-8_38. arXiv: 1907.09200.

[100] Lee, Y., Callaghan, M. F., Acosta-Cabronero, J., et al. "Establishing intra- and inter-vendor reproducibility of T 1 relaxation time measurements with 3T MRI". In: *Magnetic Resonance in Medicine* 81.1 (Jan. 2019), pp. 454–465. ISSN: 07403194. DOI: 10.1002/mrm.27421.

[101] Litjens, G., Kooi, T., Bejnordi, B. E., et al. *A survey on deep learning in medical image analysis*. Dec. 2017. DOI: 10.1016/j.media.2017.07.005.

[102] Liu, F., Xia, Y., Yang, D., et al. "An alarm system for segmentation algorithm based on shape model". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2019-Octob. Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 10651–10660. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.01075. arXiv: 1903.10645.

[103] Liu, J. and Guo, L. "An Improved K-means Algorithm for Brain MRI Image Segmentation". In: *Proceedings of the 3rd International Conference on Mechatronics, Robotics and Automation*. Paris, France: Atlantis Press, 2015. ISBN: 978-94-62520-76-9. DOI: 10.2991/icmra-15.2015.210.

[104] Loktyushin, A., Herz, K., Dang, N., et al. *MRzero – Fully automated invention of MRI sequences using supervised learning*. 2020. arXiv: 2002.04265 [physics.med-ph].

[105] Maes, F., Collignon, A., Vandermeulen, D., et al. "Multimodality image registration by maximization of mutual information". In: *IEEE Transactions on Medical Imaging* 16.2 (1997), pp. 187–198. ISSN: 02780062. DOI: 10.1109/42.563664.

[106] Masci, F. J., Makovoz, D., and Moshir, M. "A Robust Algorithm for the Pointing Refinement and Registration of Astronomical Images". In: *Publications of the Astronomical Society of the Pacific* 116.823 (Sept. 2004), pp. 842–858. ISSN: 0004-6280. DOI: 10.1086/424495.

[107] Mathur, A., Isopoussu, A., Kawsar, F., et al. "FlexAdapt: Flexible cycle-consistent adversarial domain adaptation". In: *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*. Institute of Electrical and Electronics Engineers Inc., Dec. 2019, pp. 896–901. ISBN: 9781728145495. DOI: 10.1109/ICMLA.2019.00155.

[108] Miller, K. L., Alfaro-Almagro, F., Bangerter, N. K., et al. "Multimodal population brain imaging in the UK Biobank prospective epidemiological study". In: *Nature Neuroscience* 19.11 (Oct. 2016), pp. 1523–1536. ISSN: 15461726. DOI: 10.1038/nn.4393.

[109] Mnih, V., Kavukcuoglu, K., Silver, D., et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 14764687. DOI: 10.1038/nature14236.

[110] Monari, E. and Pollok, T. "A real-time image-to-panorama registration approach for background subtraction using pan-tilt-cameras". In: *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2011*. 2011, pp. 237–242. DOI: 10.1109/AVSS.2011.6027329.

[111] Müller, R., Kornblith, S., and Hinton, G. E. "When does label smoothing help?" In: *Advances in Neural Information Processing Systems 32*. Ed. by Wallach, H., Larochelle, H., Beygelzimer, A., et al. Curran Associates, Inc., 2019, pp. 4694–4703.

[112] Murez, Z., Kolouri, S., Kriegman, D., et al. "Image to Image Translation for Domain Adaptation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Dec. 2018, pp. 4500–4509. ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00473. arXiv: 1712.00479.

[113] Naseem, I., Togneri, R., and Bennamoun, M. "Linear regression for face recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.11 (2010), pp. 2106–2112. ISSN: 01628828. DOI: 10.1109/TPAMI.2010.128.

[114] Ng, H. P., Huang, S., Ong, S. H., et al. "Medical image segmentation using watershed segmentation with texture-based region merging". In: *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS'08 - "Personalized Healthcare through Technology"*. 2008, pp. 4039–4042. ISBN: 9781424418152. DOI: 10.1109/iembs.2008.4650096.

[115] Odena, A., Dumoulin, V., and Olah, C. "Deconvolution and Checkerboard Artifacts". In: *Distill* (2016). DOI: 10.23915/distill.00003.

[116] Oktay, O., Bai, W., Guerrero, R., et al. "Stratified Decision Forests for Accurate Anatomical Landmark Localization in Cardiac Images". In: *IEEE Transactions on Medical Imaging* 36.1 (Jan. 2017), pp. 332–342. ISSN: 1558254X. DOI: 10.1109/TMI.2016.2597270.

[117] Osadebey, M., Pedersen, M., Arnold, D., et al. "Image Quality Evaluation in Clinical Research: A Case Study on Brain and Cardiac MRI Images in Multi-Center Clinical Trials". In: *IEEE*

Journal of Translational Engineering in Health and Medicine (Aug. 2018). ISSN: 21682372. DOI: 10.1109/JTEHM.2018.2855213.

[118] Otsu, N. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (Jan. 1979), pp. 62–66. ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310076.

[119] Pan, S. J. and Yang, Q. *A survey on transfer learning*. 2010. DOI: 10.1109/TKDE.2009.191.

[120] Peng, B., Zhang, L., Mou, X., et al. "Evaluation of Segmentation Quality via Adaptive Composition of Reference Segmentations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.10 (Oct. 2017), pp. 1929–1941. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2622703.

[121] Persoon, E. and Fu, K. S. "Shape Discrimination Using Fourier Descriptors". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-7.3 (1977), pp. 170–179. ISSN: 21682909. DOI: 10.1109/TSMC.1977.4309681.

[122] Petersen, S. E., Aung, N., Sanghvi, M. M., et al. "Reference ranges for cardiac structure and function using cardiovascular magnetic resonance (CMR) in Caucasians from the UK Biobank population cohort". In: *Journal of Cardiovascular Magnetic Resonance* 19.1 (Feb. 2017), p. 18. ISSN: 1532429X. DOI: 10.1186/s12968-017-0327-9.

[123] Petersen, S. E., Matthews, P. M., Francis, J. M., et al. "UK Biobank's cardiovascular magnetic resonance protocol". In: *Journal of Cardiovascular Magnetic Resonance* 18.1 (Feb. 2016), p. 8. ISSN: 1532429X. DOI: 10.1186/s12968-016-0227-4.

[124] Petersen, S. E., Sanghvi, M. M., Aung, N., et al. "The impact of cardiovascular risk factors on cardiac structure and function: Insights from the UK Biobank imaging enhancement study". In: *PLOS ONE* 12.10 (Oct. 2017). Ed. by Fukumoto, Y., e0185114. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0185114.

[125] Pohle, R. and Toennies, K. D. "Segmentation of medical images using adaptive region growing". In: *Medical Imaging 2001: Image Processing*. Ed. by Sonka, M. and Hanson, K. M. Vol. 4322. SPIE, July 2001, pp. 1337–1346. DOI: 10.1117/12.431013.

[126] Popovic, A., Fuente, M. de la, Engelhardt, M., et al. "Statistical validation metric for accuracy assessment in medical image segmentation". In: *International Journal of Computer Assisted Radiology and Surgery* 2.3-4 (July 2007), pp. 169–181. ISSN: 18616429. DOI: 10.1007/s11548-007-0125-1.

[127] Qiu, L., Li, C., and Ren, H. "Real-time surgical instrument tracking in robot-assisted surgery using multi-domain convolutional neural network". In: *Healthcare Technology Letters*. Vol. 6. 6. Institution of Engineering and Technology, 2019, pp. 159–164. DOI: 10.1049/htl.2019.0068.

[128] Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., et al. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN: 0262170051.

[129] Ramirez, P. Z., Tonioni, A., and Di Stefano, L. "Exploiting semantics in adversarial training for image-level domain adaptation". In: *IEEE 3rd International Conference on Image Processing, Applications and Systems, IPAS 2018*. Institute of Electrical and Electronics Engineers Inc., July 2018, pp. 49–54. ISBN: 9781728102474. DOI: 10.1109/IPAS.2018.8708884. arXiv: 1810.05852.

[130] Reddy, B. S. and Chatterji, B. N. "An FFT-based technique for translation, rotation, and scale-invariant image registration". In: *IEEE Transactions on Image Processing* 5.8 (1996), pp. 1266–1271. ISSN: 10577149. DOI: 10.1109/83.506761.

[131] Robinson, R., Dou, Q., Castro, D. C., et al. *Image-level Harmonization of Multi-Site Data using Image-and-Spatial Transformer Networks*. 2020. arXiv: 2006.16741 [eess.IV].

[132] Robinson, R., Oktay, O., Bai, W., et al. "Real-Time Prediction of Segmentation Quality". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11073 LNCS. Springer Verlag, Sept. 2018, pp. 578–585. ISBN: 9783030009366. DOI: 10.1007/978-3-030-00937-3_66. arXiv: 1806.06244.

[133] Robinson, R., Valindria, V. V., Bai, W., et al. "Automatic quality control of cardiac MRI segmentation in large-scale population imaging". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10433 LNCS. Springer Verlag, Sept. 2017, pp. 720–727. ISBN: 9783319661810. DOI: 10.1007/978-3-319-66182-7_82.

[134] Robinson, R., Valindria, V. V., Bai, W., et al. "Automated quality control in image segmentation: Application to the UK Biobank cardiovascular magnetic resonance imaging study". In: *Journal of Cardiovascular Magnetic Resonance* 21.1 (Mar. 2019), pp. 1–14. ISSN: 1532429X. DOI: 10.1186/s12968-019-0523-x. arXiv: 1901.09351.

[135] Rodríguez-Méndez, I. A., Ureña, R., and Herrera-Viedma, E. "Fuzzy clustering approach for brain tumor tissue segmentation in magnetic resonance images". In: *Soft Computing* 23.20 (Oct. 2019), pp. 10105–10117. ISSN: 14337479. DOI: 10.1007/s00500-018-3565-3.

[136] Rokach, L. and Maimon, O. "Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, May 2006, pp. 165–192. DOI: 10.1007/0-387-25465-x_9.

[137] Ronneberger, O., Fischer, P., and Brox, T. "U-net: Convolutional networks for biomedical image segmentation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9351. Springer Verlag, 2015, pp. 234–241. ISBN: 9783319245737. DOI: 10.1007/978-3-319-24574-4_28. arXiv: 1505.04597.

[138] Roy, A. G., Conjeti, S., Navab, N., et al. "Bayesian QuickNAT: Model uncertainty in deep whole-brain segmentation for structure-wise quality control". In: *NeuroImage* 195 (July 2019), pp. 11–22. ISSN: 10959572. DOI: 10.1016/j.neuroimage.2019.03.042. arXiv: 1811.09800.

[139] Rueckert, D., Hayes, C., Studholme, C., et al. "Non-rigid registration of breast MR images using mutual information". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 1496. Springer Verlag, Oct. 1998, pp. 1144–1152. ISBN: 3540651365. DOI: 10.1007/bfb0056304.

[140] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0.

[141] Sahba, F., Tizhoosh, H. R., and Salama, M. M. "A reinforcement learning framework for medical image segmentation". In: *IEEE International Conference on Neural Networks - Conference Proceedings*. 2006, pp. 511–517. ISBN: 0780394909. DOI: 10.1109/ijcnn.2006.246725.

[142] Sandfort, V., Yan, K., Pickhardt, P. J., et al. "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks". In: *Scientific Reports* 9.1 (Dec. 2019), pp. 1–9. ISSN: 20452322. DOI: 10.1038/s41598-019-52737-x.

[143] Schapire, R. E. "The strength of weak learnability". In: *Machine Learning* 5.2 (June 1990), pp. 197–227. ISSN: 0885-6125. DOI: 10.1007/BF00116037.

[144] Scherer, D., Müller, A., and Behnke, S. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6354 LNCS. PART 3. Springer, Berlin, Heidelberg, 2010, pp. 92–101. ISBN: 3642158242. DOI: 10.1007/978-3-642-15825-4_10.

[145] Selvaraju, R. R., Cogswell, M., Das, A., et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2

(Feb. 2020), pp. 336–359. ISSN: 15731405. DOI: 10.1007/s11263-019-01228-7. arXiv: 1610.02391.

[146] Sethian, J. A. "A fast marching level set method for monotonically advancing fronts". In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595. ISSN: 0027-8424. DOI: 10.1073/pnas.93.4.1591.

[147] Sezgin, M. and Sankur, B. "Survey over image thresholding techniques and quantitative performance evaluation". In: *Journal of Electronic Imaging* 13.1 (2004), pp. 146–165. DOI: 10.1117/1.1631315.

[148] Shafto, M. A., Tyler, L. K., Dixon, M., et al. "The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) study protocol: A cross-sectional, lifespan, multidisciplinary examination of healthy cognitive ageing". In: *BMC Neurology* 14.1 (2014). ISSN: 14712377. DOI: 10.1186/s12883-014-0204-1.

[149] Shannon, C. E. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. ISSN: 00058580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[150] Shariff, A., Kangas, J., Coelho, L. P., et al. "Automated Image Analysis for High-Content Screening and Analysis". In: *Journal of Biomolecular Screening* 15.7 (Aug. 2010). Ed. by Trask, O. J., Davies, A., and Haney, S., pp. 726–734. DOI: 10.1177/1087057110370894.

[151] Shi, W., Meng, F., and Wu, Q. "Segmentation quality evaluation based on multi-scale convolutional neural networks". In: *2017 IEEE Visual Communications and Image Processing, VCIP 2017*. Vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., Feb. 2018, pp. 1–4. ISBN: 9781538604625. DOI: 10.1109/VCIP.2017.8305140.

[152] Silverman, H. F. "A Class of Algorithms for Fast Digital Image Registration". In: *IEEE Transactions on Computers* C-21.2 (1972), pp. 179–186. ISSN: 00189340. DOI: 10.1109/TC.1972.5008923.

[153] Sobel, I. and Feldman, G. "A 3×3 isotropic gradient operator for image processing". In: *Pattern Classification and Scene Analysis* (1973), pp. 271–272.

[154] Sudlow, C., Gallacher, J., Allen, N., et al. "UK Biobank: An Open Access Resource for Identifying the Causes of a Wide Range of Complex Diseases of Middle and Old Age". In: *PLOS Medicine* 12.3 (Mar. 2015), e1001779. ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1001779.

[155] Svedlow, M., McGillem, C. D., and Anuta, P. E. "Experimental Examination of Similarity Measures and Preprocessing Methods Used for Image Registration". In: 1976.

[156] Swierczynski, P., Papież, B. W., Schnabel, J. A., et al. "A level-set approach to joint image segmentation and registration with application to CT lung imaging". In: *Computerized Medical Imaging and Graphics* 65 (Apr. 2018), pp. 58–68. ISSN: 18790771. DOI: 10.1016/j.compmedimag.2017.06.003.

[157] Taha, A. A. and Hanbury, A. "Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool". In: *BMC Medical Imaging* 15.1 (Aug. 2015), p. 29. ISSN: 14712342. DOI: 10.1186/s12880-015-0068-x.

[158] Taheri, S., Ong, S. H., and Chong, V. F. "Level-set segmentation of brain tumors using a threshold-based speed function". In: *Image and Vision Computing* 28.1 (Jan. 2010), pp. 26–37. ISSN: 02628856. DOI: 10.1016/j.imavis.2009.04.005.

[159] Talebi, H. and Milanfar, P. "NIMA: Neural Image Assessment". In: *IEEE Transactions on Image Processing* 27.8 (Aug. 2018), pp. 3998–4011. ISSN: 10577149. DOI: 10.1109/TIP.2018.2831899. arXiv: 1709.05424.

[160] Tarroni, G., Bai, W., Oktay, O., et al. "Large-scale Quality Control of Cardiac Imaging in Population Studies: Application to UK Biobank". In: *Scientific Reports* 10.1 (Dec. 2020), pp. 1–11. ISSN: 20452322. DOI: 10.1038/s41598-020-58212-2.

[161] Taylor, J. R., Williams, N., Cusack, R., et al. "The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) data repository: Structural and functional MRI, MEG, and cognitive data from a cross-sectional adult lifespan sample". In: *NeuroImage* 144 (Jan. 2017), pp. 262–269. ISSN: 10538119. DOI: 10.1016/j.neuroimage.2015.09.018. arXiv: 1910.04597.

[162] Teng, L., Li, H., and Karim, S. "DMCNN: A Deep Multiscale Convolutional Neural Network Model for Medical Image Segmentation". In: *Journal of Healthcare Engineering* 2019 (2019). Ed. by Zhao, L., p. 8597606. ISSN: 2040-2295. DOI: 10.1155/2019/8597606.

[163] Toshev, A. and Szegedy, C. "DeepPose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Sept. 2014, pp. 1653–1660. ISBN: 9781479951178. DOI: 10.1109/CVPR.2014.214. arXiv: 1312.4659.

[164] Valindria, V. V., Lavdas, I., Bai, W., et al. "Reverse Classification Accuracy: Predicting Segmentation Performance in the Absence of Ground Truth". In: *IEEE Transactions on Medical Imaging* 36.8 (Aug. 2017), pp. 1597–1606. ISSN: 0278-0062. DOI: 10.1109/TMI.2017.2665165.

[165] Vollmer, S., Mateen, B. A., Bohner, G., et al. "Machine learning and artificial intelligence research for patient benefit: 20 critical questions on transparency, replicability, ethics, and effectiveness". In: *The BMJ* 368 (Mar. 2020). ISSN: 17561833. DOI: 10.1136/bmj.l6927.

[166] Wachinger, C., Becker, B. G., Rieckmann, A., et al. "Quantifying confounding bias in neuroimaging datasets with causal inference". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11767 LNCS. Springer, July 2019, pp. 484–492. ISBN: 9783030322502. DOI: 10.1007/978-3-030-32251-9_53. arXiv: 1907.04102.

[167] Wadhwa, N., Garg, R., Jacobs, D. E., et al. "Synthetic depth-of-field with a single-camera mobile phone". In: *ACM Transactions on Graphics* 37.4 (Aug. 2018), pp. 1–13. ISSN: 15577368. DOI: 10.1145/3197517.3201329.

[168] Wang, H., Suh, J. W., Das, S. R., et al. "Multi-atlas segmentation with joint label fusion". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3 (2013), pp. 611–623. ISSN: 01628828. DOI: 10.1109/TPAMI.2012.143.

[169] Wang, K., Zhou, S., Fu, C. A., et al. "Mining Changes of Classification by Correspondence Tracing". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. Philadelphia, PA: Society for Industrial and Applied Mathematics, May 2003, pp. 95–106. ISBN: 978-0-89871-545-3. DOI: 10.1137/1.9781611972733.9.

[170] Wang, Z., Bovik, A. C., Sheikh, H. R., et al. "Image quality assessment: From error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. ISSN: 10577149. DOI: 10.1109/TIP.2003.819861.

[171] Warfield, S. K., Zou, K. H., and Wells, W. M. "Simultaneous truth and performance level estimation (STAPLE): An algorithm for the validation of image segmentation". In: *IEEE Transactions on Medical Imaging* 23.7 (July 2004), pp. 903–921. ISSN: 02780062. DOI: 10.1109/TMI.2004.828354.

[172] Widmer, G. and Kubat, M. "Learning in the presence of concept drift and hidden contexts". In: *Machine Learning* 23.1 (Apr. 1996), pp. 69–101. ISSN: 0885-6125. DOI: 10.1007/BF00116900.

[173] Willemink, M. J., Koszek, W. A., Hardell, C., et al. *Preparing medical imaging data for machine learning*. Feb. 2020. DOI: 10.1148/radiol.2020192224.

[174] Wilson, G. and Cook, D. J. "A Survey of Unsupervised Deep Domain Adaptation". In: (Dec. 2018). arXiv: 1812.02849.

[175] Wu, Y., Duan, H., and Du, S. "Multiple fuzzy c-means clustering algorithm in medical diagnosis". In: *Technology and Health Care*. Vol. 23. s2. IOS Press, June 2015, S519–S527. DOI: 10.3233/THC-150989.

[176] Yan, W., Wang, Y., Gu, S., et al. "The Domain Shift Problem of Medical Image Segmentation and Vendor-Adaptation by Unet-GAN". In: (Oct. 2019). arXiv: 1910.13681.

[177] Yang, J., Dvornek, N. C., Zhang, F., et al. "Unsupervised Domain Adaptation via Disentangled Representations: Application to Cross-Modality Liver Segmentation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11765 LNCS. Springer, Oct. 2019, pp. 255–263. ISBN: 9783030322441. DOI: 10.1007/978-3-030-32245-8_29. arXiv: 1907.13590.

[178] Zhang, L., Gooya, A., Dong, B., et al. "Automated quality assessment of cardiac MR images using convolutional neural networks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9968 LNCS. Springer Verlag, Oct. 2016, pp. 138–145. ISBN: 9783319466293. DOI: 10.1007/978-3-319-46630-9_14.

[179] Zhang, R. and Chung, A. C. "A Fine-Grain Error Map Prediction and Segmentation Quality Assessment Framework for Whole-Heart Segmentation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11765 LNCS. Springer, Oct. 2019, pp. 550–558. ISBN: 9783030322441. DOI: 10.1007/978-3-030-32245-8_61. arXiv: 1907.12244.

[180] Zhang, X., Tian, J., Xiang, D., et al. "Interactive liver tumor segmentation from ct scans using support vector classification with watershed". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. 2011, pp. 6005–6008. ISBN: 9781424441211. DOI: 10.1109/IEMBS.2011.6091484.

[181] Zhang, Z., Wang, M., Yang, Z., et al. "Noninvasive prediction of pulmonary artery pressure and vascular resistance by using cardiac magnetic resonance indices". In: *International Journal of Cardiology* 227 (Jan. 2017), pp. 915–922. ISSN: 18741754. DOI: 10.1016/j.ijcard.2016.10.068.

[182] Zhong, E., Fan, W., Yang, Q., et al. "Cross Validation Framework to Choose amongst Models and Datasets for Transfer Learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6323 LNAI. PART 3. 2010, pp. 547–562. ISBN: 3642159389. DOI: 10.1007/978-3-642-15939-8_35.

[183] Zhou, L., Deng, W., and Wu, X. "Robust Image Segmentation Quality Assessment without Ground Truth". In: (Mar. 2019). arXiv: 1903.08773.

[184] Zhu, J. Y., Park, T., Isola, P., et al. "Unpaired Image-to-Image Translation Using Cycle Consistent Adversarial Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2017-Octob. Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 2242–2251. ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.244. arXiv: 1703.10593.

[185] Zikic, D., Glocker, B., and Criminisi, A. "Encoding atlases by randomized classification forests for efficient multi-atlas label propagation". In: *Medical Image Analysis* 18.8 (Dec. 2014), pp. 1262–1273. ISSN: 13618415. DOI: 10.1016/j.media.2014.06.010.

[186] Zitová, B. and Flusser, J. "Image registration methods: A survey". In: *Image and Vision Computing* 21.11 (Oct. 2003), pp. 977–1000. ISSN: 02628856. DOI: 10.1016/S0262-8856(03)00137-9.

[187] Zou, K. H., Warfield, S. K., Bharatha, A., et al. "Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index". In: *Academic Radiology* 11.2 (Feb. 2004), pp. 178–189. ISSN: 10766332. DOI: 10.1016/S1076-6332(03)00671-8.

Appendices

The GitHub repository at https://github.com/mlnotebook contains the code used in this work as well as those snippets shown here.

Listing 1: Segmentation by Thresholding in Python.

Listing 2: Segmentation by Gaussian Mixture Model (GMM) in Python.

Listing 3: Segmentation by Edge Detection in Python.

Listing 4: Mutual Information Visualization in Python.

Listing 5: Composite Affine Matrix Construction and Interpolation Demonstration in Python.

Listing 6.1: Segmentation by Thresholding in Python. Python code to perform image segmentation via basic thresholding.

```
import argparse
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt

def threshold_it(image, thresholds, output_dir, segments=False):
   ### Successively applies each threshold in `threshold` to the given 2D grayscale `image`.
   ### Saves the result of each threshold step into `output_dir`.
   # Args.
   #    image: numpy array of floats or ints - single channel 2D image to be thresholded with shape (H, W).
   #    threhsolds: list - list of intensities to be used as threhsolds.
   #   output_dir: str - directory in which to store the intermediate outputs if `segments` is True.
   #   segments: bool - if True, the segmentation result at each threshold is saved.
   # Returns.
   #    thresholded_image: numpy array of ints - 2D image.

   assert(type(thresholds) is list), "thresholds should be a list"
   assert((len(image.shape) == 2) and (type(image) is np.ndarray)), "image should be a 2D numpy array"

   thresholded_image = np.zeros(image.shape).astype(np.int32)

   for threshold in thresholds:
         thresholded_image[np.where(image > threshold)] += 1

         if segments:
            this_segment = np.zeros(image.shape)
            this_segment[np.where(image > threshold)] = 1
            plt.imshow(this_segment, cmap='gray')
            plt.axis('off')
            plt.tight_layout()
            filename = os.path.join(output_dir, 'thresholded_{}.png'.format(threshold))
            plt.savefig(filename, dpi=200)

   return thresholded_image


if __name__ == "__main__":
   parser = argparse.ArgumentParser(description = 'SegmentationThreshold')
   parser.add_argument('--image', default="./gmm_brain.png", help='str: Path to the image to be segmented.')
   parser.add_argument('--thresholds', nargs="+", default=[4, 100, 180], help='space separated list: A space separated list of intensity values to use for thresholding.')
   parser.add_argument('--output', default='./', help='str: Path to the output directory.')
   parser.add_argument('--segments', action='store_true', help='bool: If True, will output individual segments at each given threshold, otherwise only the final segmentation is shown.')

   # Parse arguments
   args = parser.parse_args()
   image_path = args.image
   thresholds = [float(x) for x in args.thresholds]
   segments = args.segments
   output_dir = args.output
```

```
51
52      # Create the output dir if it doesn't exist
53      os.makedirs(output_dir, exist_ok=True)
54
55      # Read in the image
56      assert(os.path.exists(image_path)), "The specified image does not exist at
        ↪  {}".format(args.image)
57      img = cv2.imread(args.image, cv2.IMREAD_UNCHANGED)
58
59      # Perform thresholding
60      thresholded_image = threshold_it(img, thresholds, output_dir, segments)
61
62      # Save the result
63      plt.imshow(thresholded_image, cmap='gray')
64      plt.axis('off')
65      plt.tight_layout()
66      plt.savefig(os.path.join(output_dir, 'thresholded.png'), dpi=200)
```

Listing 6.2: Segmentation by Gaussian Mixture Model (GMM) in Python. Python code to perform image segmentation with Gaussian Mixture Models.

```
import argparse
import cv2
import os
import numpy as np
import SimpleITK as sitk
from matplotlib import cm
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture

def normalize_0_255(array):
   ### Function to normalize the values of an n-dimensional array
   ### such that they fall in the range [0, 255].
   # Args.
   #    array: numpy array of floats or ints - the n-dimensional
   #         array to be normalized.
   # Returns.
   #    array: numpy array of floats - array of equal shape to
   #         the input. Values in the range [0, 255].

   assert(type(array) is np.ndarray), "Input should be numpy.ndarray"

   array = array.astype(np.float32) - array.min()
   array = array / array.max()
   array = array * 255.
   return array

if __name__ == '__main__':
   parser = argparse.ArgumentParser(description = 'SegmentationGausianMixtureModel')
   parser.add_argument('--image', default="./gmm_brain.nii.gz", help='str: Path to the image to
   ↪  be segmented.')
   parser.add_argument('--output', default='./', help='str: Path to the output directory.')
   parser.add_argument('--n', nargs="+", default=[1, 2, 3, 4, 5, 6], help='space-separated list
   ↪  of ints: Each value is passed as n_components for GMM. One GMM is generated for each value
   ↪  in the list.')
   parser.add_argument('--modes', action='store_true', help='bool: If True, the individual models
   ↪  that comprise the GMM are also plotted on the histogram.')

   # Parse arguments
   args = parser.parse_args()
   image_path = args.image
   output_dir = args.output
   n_components = [int(x) for x in args.n]
   plot_modes = args.modes

   # Create the output dir if it doesn't exist
   os.makedirs(output_dir, exist_ok=True)

   # Read in the image
   assert(os.path.exists(image_path)), "The specified image does not exist at
   ↪  {}.".format(args.image)
   assert(os.path.basename(image_path).split('.', 1)[-1] in ['nii.gz', '.png', '.jpg']), "Image
   ↪  should be .nii.gz, .png or .jpg format."

   ### If '.nii.gz' file given acquire, normalize and extract central slice as .png
   if image_path.endswith('.nii.gz'):
      image = sitk.ReadImage(image_path)
      image = sitk.GetArrayFromImage(image)
```

```
    H, W, D = image.shape

    image = image[H//2, ::-1, ::-1] #flip to correct orientation
    image = normalize_0_255(image)
    image_path = os.path.join(output_dir, 'gmm_brain.png')
    cv2.imwrite(image_path, image.astype(np.uint8))

### Read in the image file, and produce histogram.
img = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)

hist = cv2.calcHist([img], [0], None, [256], [0, 256])
data = np.ravel(img)

# Remove the very high count of background pixels
hist[0] = 0
data = data[data != 0]

# Plot image histogram
x = np.linspace(0, 254, 256)
fig, ax = plt.subplots()
ax.hist(img.ravel(), 255, [2,256], density=True)
plt.ylabel("Frequency", fontsize=14)
plt.xlabel("Pixel Intensity", fontsize=14)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'brain_hist.png'), dpi=200)
plt.close()

### Fit GMM to the image histogram for each number of Gaussians specified in n_components
for n in n_components:
    cmap = cm.get_cmap('tab20', n)
    gmm = GaussianMixture(n_components = n)
    gmm = gmm.fit(X=data.reshape(-1,1))

    # Evaluate the model
    gmm_x = np.linspace(0, 254, 256) #removed the 0 (background) bin earlier
    gmm_y = np.exp(gmm.score_samples(gmm_x.reshape(-1,1)))

    # Plot histogram and overall GMM
    fig, ax = plt.subplots()
    ax.hist(np.ravel(img), 255, [2,256], density=True)
    ax.plot(gmm_x, gmm_y, color="Red", lw=3)


    # Plot each GMM mode
    if plot_modes:
        for i in range(n):
            normalization = 1 / (np.sqrt(gmm.covariances_[i]) * np.sqrt(2*np.pi))
            exponent = -0.5 * ((gmm_x.reshape(-1,1) - gmm.means_[i]) /
            ↪  np.sqrt(gmm.covariances_[i]))**2
            i_y = gmm.weights_[i] * normalization * np.exp(exponent)
            ax.plot(gmm_x, i_y, "k--")

    plt.tick_params(axis='y', which='both', left=False, labelleft=False)
    plt.xlabel("Pixel Intensity", fontsize=14)
    plt.tight_layout()
    plt.savefig(os.path.join(output_dir, 'gmm_{}.png'.format(n)), dpi=200)
    plt.close()

    # Segment image
    clusters = gmm.predict(img.reshape(-1,1))
    clusters = clusters.reshape(img.shape)
    plt.imshow(clusters, cmap=cmap)
```

```
        plt.axis('off')
        plt.tight_layout()
        plt.savefig(os.path.join(output_dir, 'gmm_seg_{}.png'.format(n)), dpi=200)
```

Listing 6.3: Segmentation by Edge Detection in Python. Python code to perform image segmentation via edge detection.

```
import argparse
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt

def gaussian(x, mu, sigma):
   ### Returns the value of the Gaussian distribution with mean `mu` and standard deviation `sigma` at position `x`.
   ## Args.
   #   x - float - value to be evaluated.
   #    mu - float - mean of the Gaussian distribution.
   #    sigma - float - standard deviation of the Gaussian distribution.
   ## Returns.
   #    gaussian - float - value of the Gaussian distribution with mean `mu` and standard deviation `sigma` at position `x`.

   exponent = -0.5*((x - mu)/sigma)**2
   normalizer = sigma * np.sqrt(2 * np.pi)
   gaussian = normalizer * np.exp(exponent)

   return gaussian


def create_kernel_dict(blur_size=5, blur_mean=0, blur_sigma=1):
   ### Creates a dictionary of kernels (2D numpy arrays) including: Gaussian blurring kernel, sobel edge filters (x and y) and laplacian edge filter.
   ## Args.
   #    blur_size - int - the height (and width) of the Gaussian blurring kernel.
   #   blur_mean - float - mean of the Gaussian distribution over which the Gaussian blurring kernel is calcualted.
   #   blur_sigma - float - standard deviation of the Gaussian distribution over which the Gaussian blurring kernel is calcualted.
   ## Returns.
   #    kernel_dict - dictionary{str: numpy array} - a dictionary of str keys and numpy array values.

   gaussian_kernel_1D = [gaussian(x, blur_mean, blur_sigma) for x in range(-(blur_size-1)//2, (blur_size+1)//2)]
   gaussian_kernel_2D = np.outer(gaussian_kernel_1D, gaussian_kernel_1D)

   laplacian = np.array(([0, 1, 0], [1, -4, 1], [0, 1, 0]), dtype=np.float32)
   sobel_x = np.array(([-1, 0, 1], [-2, 0, 2], [-1, 0, 1]), dtype=np.float32)
   sobel_y = np.array(([-1, -2, -1], [0, 0, 0], [1, 2, 1]), dtype=np.float32)

   kernel_dict = {'laplacian': laplacian,
      'sobel_x': sobel_x,
      'sobel_y':sobel_y,
      'gaussian_blur': gaussian_kernel_2D}

   return kernel_dict

def pad_image(image, kernel):
   ### Performs padding of `image` such that convolution of `kernel` with the `image`
   ## Args.
   #    image - numpy array - a 2D numpy array of shape (IH, IW).
   #   kernel - numpy array - a 2D numpy array of shape (KH, KW) where KH < IH and KW < IW.
```

```
    ## Returns.
    #    padded_image - numpy array - a 2D numpy array with `image` occupying the central area.

    assert((len(image.shape) == 2) and (type(image) is np.ndarray)), "Image must be a 2D numpy array."
    assert((len(kernel.shape) == 2) and (type(kernel) is np.ndarray)), "Kernel must be a 2D numpy array."
    assert((kernel.shape[0] < image.shape[0]) and (kernel.shape[1] < image.shape[1])), "Kernel must be smaller than the image."

    img_h, img_w = image.shape
    k_h, k_w = kernel.shape

    pad_h = int((k_h - 1) / 2)
    pad_w = int((k_w - 1) / 2)

    padded_image = np.zeros([img_h + (2*pad_h), img_w + (2*pad_w)])
    padded_image[pad_h:pad_h + img_h, pad_w: pad_w + img_w] = image

    return padded_image

def convolution(image, kernel):
    ### Performs 2-dimensional convolution of the `kernel` with the `image`.
    ## Args.
    #    image - numpy array - a 2D numpy array of shape (IH, IW).
    #   kernel - numpy array - a 2D numpy array of shape (KH, KW) where KH < IH and KW < IW.
    ## Returns.
    #    output - nump.ndarray - a 2D numpy array with the same shape as `image`.

    assert((len(image.shape) == 2) and (type(image) is np.ndarray)), "Image must be a 2D numpy array."
    assert((len(kernel.shape) == 2) and (type(kernel) is np.ndarray)), "Kernel must be a 2D numpy array."
    assert((kernel.shape[0] < image.shape[0]) and (kernel.shape[1] < image.shape[1])), "Kernel must be smaller than the image."

    img_h, img_w = image.shape
    k_h, k_w = kernel.shape

    image = pad_image(image, kernel)
    output = np.zeros(image.shape)

    for row in range(img_h):
        for col in range(img_w):
            output[row, col] = np.sum(kernel * image[row:row + k_h, col:col + k_w])
            output[row, col] /= kernel.shape[0] * kernel.shape[1]

    return output


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description = 'SegmentationEdgeDetection')
    parser.add_argument('--image', default="./gmm_brain.png", help='str: Path to the image to be segmented.')
    parser.add_argument('--output', default='./', help='str: Path to the output directory.')

    # Parse arguments
    args = parser.parse_args()
    image_path = args.image
    output_dir = args.output

    # Create the output dir if it doesn't exist
```

```
    os.makedirs(output_dir, exist_ok=True)

    # Read in the image
    assert(os.path.exists(image_path)), "The specified image does not exist at
    ↪  {}.".format(args.image)
    img = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)

    kernels = create_kernel_dict()
    for kernel in kernels.keys():
        filtered_img = convolution(img, kernels[kernel])

        fig, ax = plt.subplots()
        ax.imshow(filtered_img, cmap='gray')

        ax.set_axis_off()
        plt.tight_layout()
        plt.savefig(os.path.join(output_dir, 'filtered_{}.png'.format(kernel)))
        plt.close()

    filtered_img = convolution(img, kernels['gaussian_blur'])
    filtered_img = convolution(filtered_img, kernels['laplacian'])

    fig, ax = plt.subplots()
    ax.imshow(filtered_img, cmap='gray')
    ax.set_axis_off()

    plt.tight_layout()
    plt.savefig(os.path.join(output_dir, 'filtered_blurred_laplacian.png'))
    plt.close()
```

Listing 6.4: Mutual Information Visualization in Python. Visualization of the MI metric used for intensity-based registration, *e.g.* template-matching.

```
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib import gridspec
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter)
from scipy.stats import gaussian_kde


def plot_image_histogram(images, title, savename, output_dir='./', nbins=15, roi=None, lims=[0, 255], kde=True):
   ### Plots and saves a 1D historgam of intensities for 1 image, or a joint 2D histogram of intensities for 2 images.
   ## Args.
   #   images: list np.array floats - list of images whose histograms are to be calculated.
   #      If len(images) == 1, a 1D histgram of intensity counts is plotted.
   #      If len(images) == 2, a 2D joint histogram of intensity counts is plotted.
   #         image list should be [[image]] or [[template], [window]].
   #   title: str- title for the image
   #   savename: str - name for the output file
   #   output_dir: str - the root directory in which to save figures.
   #   nbins: number of bins to split the histogram into.
   #   lims: list float [min,max] - the intensity lower and upper bounds. Default = [0, 255]
   #   kde: bool - whether to plot kde over histogram

   assert (type(images) == list) & (0 < len(images) <= 2), 'images should be a list of length 1 or 2'

   roi_label = ''
   if roi:
      roi_label = '_ROI_[{}]'.format(','.join([str(r) for r in roi]))
      title = title + '\n{}'.format(roi_label[1:])

   if len(images) == 1:
      img = images[0]
      fig, ax = plt.subplots(figsize=(5,5))
      h, w = img.shape
      im1 = img.reshape(-1)
      ax.hist(im1, bins=nbins, density=True, range=lims)

      if False:
         kde_im1 = gaussian_kde(im1)
         x = np.linspace(lims[0], lims[1], 100)
         dx = kde_im1(x)
         ax.plot(dx, x, c='black', lw=1)

      ax.set_title(title, fontsize=18)
      fig.tight_layout()
   else:
      gs = gridspec.GridSpec(2, 2, width_ratios=[5,1], height_ratios = [1, 5], wspace=0.01, hspace=0.01)
      fig = plt.figure(figsize=(8,8))

      im1 = images[0].reshape(-1)
      im2 = images[1].reshape(-1)

```

```
        #Plot the 2D histogram and scatter plot on top of each other - note "extent" in imshow.
        ax = plt.subplot(gs[1, 0], xlim = (lims[0], lims[1]), ylim = (lims[0], lims[1]))
        hist_2d, xs, ys = np.histogram2d(im1, im2, bins=nbins, range=[lims,lims])
        hist_2d[hist_2d!=0] = np.log(hist_2d[hist_2d!=0])
        histogram_handle = ax.imshow(hist_2d.T, origin='lower', cmap='gray', extent=[0,255,0,255])
        scatter_handle = ax.scatter(im1, im2, color='deepskyblue', alpha=0.6)

        # Set the axis and grid parameters for the scatter plot
        minor_multiple = (lims[1] - lims[0]) / nbins
        major_multiple = nbins*2

        ax.xaxis.set_major_locator(MultipleLocator(major_multiple))
        ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
        ax.xaxis.set_minor_locator(MultipleLocator(minor_multiple))
        ax.yaxis.set_major_locator(MultipleLocator(major_multiple))
        ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
        ax.yaxis.set_minor_locator(MultipleLocator(minor_multiple))
        ax.grid(True, 'minor', 'both', lw=0.5, c='gray', ls='--')

        #Plot the histogram for the window on the right
        right_ax = plt.subplot(gs[1, 1], ylim = (lims[0], lims[1]), frameon=False)
        right_ax.hist(im2, range=lims, color = 'deepskyblue', bins=nbins, orientation =
        ↪  'horizontal', density=kde)
        right_ax.grid(True, 'minor', 'y', lw=0.5, c='gray', ls='--')
        #Plot the histogram for the template on the right
        top_ax = plt.subplot(gs[0,0], xlim = (lims[0], lims[1]), frameon=False)
        top_ax.hist(im1, range=lims, color = 'deepskyblue', bins=nbins, density=kde)
        top_ax.grid(True, 'minor', 'x', lw=0.5, c='gray', ls='--')

        if kde:
           kde_im1 = gaussian_kde(im1)
           kde_im2 = gaussian_kde(im2)
           x= np.linspace(lims[0], lims[1], 100)
           y= np.linspace(lims[0], lims[1], 100)
           dx=kde_im1(x)
           dy=kde_im2(y)
           right_ax.plot(dy, y, c='black', lw=1)
           top_ax.plot(x, dx, c='black', lw=1)

        #Remove all ticks and labels from the histograms
        right_ax.yaxis.set_minor_locator(MultipleLocator(minor_multiple))
        right_ax.yaxis.set_major_locator(MultipleLocator(major_multiple))
        top_ax.xaxis.set_minor_locator(MultipleLocator(minor_multiple))
        top_ax.xaxis.set_major_locator(MultipleLocator(major_multiple))

        for axi in (right_ax.xaxis, right_ax.yaxis, top_ax.xaxis, top_ax.yaxis):
           for tic in axi.get_major_ticks() + axi.get_minor_ticks():
              tic.tick1line.set_visible(False)
              tic.tick2line.set_visible(False)
              tic.label1.set_visible(False)
              tic.label2.set_visible(False)


        ax.set_xlabel('Template Intensity', fontsize=14)
        ax.set_ylabel('Window Intensity', fontsize=14)

    plt.savefig(os.path.join(output_dir, '{}{}.png'.format(savename, roi_label)), dpi=200)
    plt.close()



def plot_image(img, title, savename, output_dir='./', rois=None):
```

```
    ### Plots and saves figures on datapoints x, y
    ## Args.
    #   img: np.array floats - image to be shown
    #   title: str- title for the image
    #   savename: str - name for the output file
    #   output_dir: str - the root directory in which to save figures.
    h, w = img.shape
    fig, ax = plt.subplots(figsize=(5,5))

    ax.imshow(img, cmap='gray', interpolation='none')
    roi_labels = []

    if rois:
        assert (type(rois)==list) & (type(rois[0]) == list), "roi should be a list of ROIs, e.g. [[x1, y1, x2, y2]]"
        colors = ['lime', 'cyan', 'red']
        roi_handles = []
        for i, roi in enumerate(rois):
            assert len(roi) == 4, "ROI should be: [x1, y1, x2, y2] where (x1,y1) = bottom-left, (x2,y2) = top-right"
            x1, y1, x2, y2 = roi
            roi_labels.append('ROI_{}_[{},{},{},{}]'.format(i, x1, y1, x2, y2))
            roi_handles.append(patches.Rectangle((x1,y1), x2-x1, y2-y1, linewidth=1, edgecolor=colors[i], facecolor='none'))
            ax.add_patch(roi_handles[-1])
        plt.legend(roi_handles, roi_labels, loc='lower right')

    plt.axis('off')
    ax.set_title(title, fontsize=18)
    plt.tight_layout()
    roi_string = '_' + '_'.join([roi_label for roi_label in roi_labels]) if roi_labels else ''
    plt.savefig(os.path.join(output_dir, '{}{}.png'.format(savename, roi_string)), dpi=200)
    plt.close()

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='AffineTransformations')
    parser.add_argument('--image', default="./gmm_brain.png", help='str: Path to the image to be segmented.')
    parser.add_argument('--output', default='./', help='str: Path to the output directory.')

    # Parse arguments
    args = parser.parse_args()
    image_path = args.image
    output_dir = args.output

    # Create the output dir if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    # Read in the image
    assert(os.path.exists(image_path)), "The specified image does not exist at {}".format(args.image)
    img = cv2.imread(args.image, cv2.IMREAD_UNCHANGED)
    h, w = img.shape

    plot_image_histogram([img], 'Original Image', 'original_image_histogram', output_dir=output_dir, nbins=20)

    # Set a template region
    tx1, ty1, tx2, ty2 = 86, 46, 106, 66
    template_roi = [tx1, ty1, tx2, ty2]
    template = img[ty1:ty2,tx1:tx2]

```

```
    wx1, wy1, wx2, wy2 = tx1+5, ty1+5, tx2+5, ty2+5
    window_roi = [wx1, wy1, wx2, wy2]
    window = img[wy1:wy2,wx1:wx2]

    nbins = 15
    plot_image(img, 'Original Image'.format(roi_title = ''),
       'original_image_roi',
       output_dir=output_dir,
       rois=[template_roi,window_roi])

    plot_image(template, 'Template',
       'template',
       output_dir=output_dir)
    plot_image(window, 'Window',
       'window',
       output_dir=output_dir,
       rois=[window_roi])

    plot_image_histogram([window], 'Window',
       'window_hist',
       output_dir=output_dir,
       nbins=nbins,
       roi=window_roi)
    plot_image_histogram([template], 'Template Histogram',
       'template_histogram',
       output_dir=output_dir,
       nbins=nbins,
       roi=template_roi)
    plot_image_histogram([window, template], 'Joint Template-Window Histogram',
       'original_image_histogram',
       output_dir=output_dir,
       nbins=nbins,
       roi=window_roi)

    wx1, wy1, wx2, wy2 = tx1, ty1, tx2, ty2
    window_roi = [wx1, wy1, wx2, wy2]
    window = img[wy1:wy2,wx1:wx2]

    plot_image(img, 'Original Image',
       'original_image_roi',
       output_dir=output_dir,
       rois=[template_roi,window_roi])

    plot_image(window, 'Window',
       'window',
       output_dir=output_dir,
       rois=[window_roi])
    plot_image_histogram([window], 'Window',
       'window_hist',
       output_dir=output_dir,
       nbins=nbins,
       roi=window_roi)
    plot_image_histogram([window, template], 'Joint Template-Window Histogram',
       'original_image_histogram',
       output_dir=output_dir,
       nbins=nbins,
       roi=window_roi)


```

Listing 6.5: Composite Affine Matrix Construction and Interpolation Demonstration in Python. Creates composite affine transformation matrix from translation, rotation and scaling matrices. Demonstrates the effect of nearest-neighbours and bilinear interpolation. Displays the warped image grid.

```
import argparse
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from scipy.interpolate import griddata

def translation_matrix(tx, ty):
   """ Creates an affine matrix with translation componenets.
   Args:
      tx: (float) - number of pixels to shift in the x-direction.
      ty: (float) - number of pixels to shift in the y-direction.
   Returns:
      T: (np.array of floats) - affine transformation matrix.
   """

   T = np.array([[1, 0, tx], [0, 1, ty], [0, 0, 1]])
   return T

def rotation_matrix(theta):
   """ Creates an affine matrix with rotation componenets.
   Args:
      theta: (float) - the counter-clockwise rotation angle (from x-axis) in radians.
   Returns:
      T: (np.array of floats) - affine transformation matrix.
   """
   theta = np.radians(theta)
   R = np.array([[np.cos(theta), np.sin(theta), 0], [-np.sin(theta), np.cos(theta), 0], [0, 0,
   ↪  1]])
   return R

def scaling_matrix(sx, sy):
   """ Creates an affine matrix with scaling componenets.
   Args:
      sx: (float) - factor by which to scale the x-axis.
      sy: (float) - factor by which to scale the y-axis.
   Returns:
      T: (np.array of floats) - affine transformation matrix.
   """
   S = np.array([[sx, 0, 0], [0, sy, 0], [0, 0, 1]])
   return S

def normit(img):
   """ Scales the image intensities to fall in the range [-1, 1].
   Args:
      img: (np.array) - the image to be normalized.
   Returns:
      img_: (np.array of floats) - the normalized image.
   """
   img_ = img - np.min(img)
   img_ = img / np.max(img_)
   return img_

```

```
def plotit(img, title, savename, output_dir='./', zoom=[]):
    """ Plots and saves figures on datapoints x, y
    Args.
        img: np.array floats - image to be shown
        title: str- title for the image
        savename: str - name for the output file
        output_dir: str - the root directory in which to save figures.
    """

    h, w = img.shape
    fig, ax = plt.subplots(figsize=(5,5))

    img = normit(img)

    if zoom:
        assert (type(zoom)==list) & (len(zoom) == 4), "zoom should be a list of 4 values [top-left-x, top-left-y, height, width]"
        assert (zoom[-2] < h) & (zoom[-1] < w), "zoom region should be less than image size"

        zx, zy, zh, zw = zoom
        patch = img[zy:zy+zh, zx:zx+zw]
        patch = cv2.resize(patch, (w//2, w//2), interpolation=cv2.INTER_NEAREST)

        img[h-patch.shape[1]:h, w - patch.shape[0]:w] = np.ones_like(patch)
        edge = 2
        patch = patch[edge:-edge, edge:-edge]
        img[h-patch.shape[1] - edge:h-edge, w-patch.shape[0] - edge:w-edge] = patch

        roi = patches.Rectangle((zx,zy),zw,zh,linewidth=2,edgecolor='w',facecolor='none')

    ax.imshow(img, cmap='gray', interpolation='none')
    ax.axhline(h//2, color='lime', lw=0.75)
    ax.axvline(w//2, color='lime', lw=0.75)

    if zoom:
        ax.add_patch(roi)

    plt.axis('off')
    ax.set_title(title, fontsize=18)
    plt.tight_layout()
    plt.savefig(os.path.join(output_dir, '{}.png'.format(savename)), dpi=200)
    plt.close()

def create_grid(h, w):
    """ Creates a mesh-grid of 2D points of shape [h, w] and returns their co-ordinates as
    stacked arrays of 1D values.
    Args:
        h: (int) - the number of points in the y-direction (height).
        w: (int) - the number of points in the x-direction (width).
    Returns:
        mesh_grid: (np.array of ints) - the x and y co-ordinates of the mesh grid.
    """
     grid = np.mgrid[0:h,0:w].reshape(2, -1)
     return np.vstack([grid, np.ones(grid.shape[1])]).astype(np.int)

def create_affine_matrix(O, transforms):
    """ Creates the composite affine matrix from origin-shift transform and `transforms`. Origin shift
    ensures that rotations of the input are computed as if it were placed at the origin.
    Args:
        O: (np.array floats) - the translation transformation to shift the input to the origin.
        transforms: (list of np.array) - the transformation matrices to apply. Taken in order.
```

```
    Returns:
        affine_matrix: (np.array) - the composite affine transformation matrix.
    """
    A = np.linalg.inv(O)
    for transform in transforms:
        A = transform @ A
    return O @ A

def get_transformed_image(img, grid, transformed_grid, interpolation=0):
    """ Resamples an image to the transformed grid.
    Args:
        img: (np.array) - the original untransformed image.
        grid: (list of list of floats) - the untransformed mesh grid as a list of x and y coordinates
            [[x0, x1 ...], [y0, y1, ...]].
        transformed_grid: (list of list of floats) - the TRANSFORMED mesh grid to be resampled as a
↪  list
            of x and y coordinates [[x0, x1 ...], [y0, y1, ...]].
        interpolation: (int) - order of the interpolation to be applied during resampling.
            0 = no interpolation, 1 = nearest neighbour, 2 = linaer.
    Returns:
        transformed_image: (np.array) - the original image resampled onto the transformed grid.
    """
    h, w = img.shape

    imgx, imgy = grid[0], grid[1]
    transx, transy = transformed_grid[0], transformed_grid[1]

    if interpolation:
        #Regrid the data with interpolation
        transformed_image = griddata((transx,transy),img.reshape(-1),(imgx,imgy),
        ↪  method=interpolation, fill_value=0)
        transformed_image = transformed_image.reshape(h,w)
    else:
        idx = (transx >=0) & (transx < h) & (transy >=0) & (transy < w)
        transx_in_x, transy_in_y = transx[idx], transy[idx]
        img_x, img_y = imgx[idx], imgy[idx]

        canvas = np.zeros_like(img)
        canvas[transx_in_x, transy_in_y] = img[img_x, img_y]
        transformed_image = canvas

    return transformed_image


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='AffineTransformations')
    parser.add_argument('--image', default="./gmm_brain.png", help='str: Path to the image to be
    ↪  segmented.')
    parser.add_argument('--output', default='./', help='str: Path to the output directory.')

    # Parse arguments
    args = parser.parse_args()
    image_path = args.image
    output_dir = args.output

    # Create the output dir if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    # Read in the image
    assert(os.path.exists(image_path)), "The specified image does not exist at
    ↪  {}".format(args.image)
    img = cv2.imread(args.image, cv2.IMREAD_UNCHANGED)
```

```
    h, w = img.shape

    plotit(img, 'Original Image', 'original', output_dir)

    interpolations = {0: 'None', 'nearest': 'NN', 'linear': 'Bilinear'}

    O = translation_matrix(h//2, w//2)

    translations = [-10, 20]
    for ty in translations:
        for tx in translations:
            grid = create_grid(h, w)
            T = translation_matrix(tx, ty)
            A = create_affine_matrix(O, [T])
            transformed_grid = np.round(A @ grid).astype(np.int)
            img_ = get_transformed_image(img, grid, transformed_grid)
            plotit(img_,
                "Translation: $t_x$={}, $t_y$={}".format(tx, ty),
                "translation_tx_{}_ty_{}".format(tx, ty),
                output_dir)

    rotations = [30, 80]
    for theta in rotations:
        for interpolation in interpolations.keys():
            grid = create_grid(h, w)
            T = rotation_matrix(theta)
            A = create_affine_matrix(O, [T])
            transformed_grid = np.round(A  @ grid).astype(np.int)
            transformed_image = get_transformed_image(img, grid, transformed_grid, interpolation)
            plotit(transformed_image,
                "Rotation: $\\theta={}$\nInterpolation: {}".format(theta,
                ↪  interpolations[interpolation]),
                "rotation_theta_{}_{}".format(theta, interpolations[interpolation]),
                output_dir)

    rotations = [30]
    zoom = [50,50,40,40]
    for theta in rotations:
        for interpolation in interpolations.keys():
            grid = create_grid(h, w)
            T = rotation_matrix(theta)
            A = create_affine_matrix(O, [T])
            transformed_grid = np.round(A  @ grid).astype(np.int)
            transformed_image = get_transformed_image(img, grid, transformed_grid, interpolation)
            plotit(transformed_image,
                "Rotation: $\\theta={}$\nInterpolation: {}".format(theta,
                ↪  interpolations[interpolation]),
                "rotation_theta_{}_{}_zoom_{}".format(theta, interpolations[interpolation], zoom),
                output_dir, zoom=zoom)

    scales = [(0.5,0.5), (1.5,0.5)]
    for scale in scales:
        for interpolation in interpolations.keys():
            grid = create_grid(h, w)
            T = scaling_matrix(scale[0], scale[1])
            A = create_affine_matrix(O, [T])
            transformed_grid = np.round(A  @ grid).astype(np.int)
            transformed_image = get_transformed_image(img, grid, transformed_grid, interpolation)
            plotit(transformed_image,
                "Scaling: $s_x={}, s_y={}$\nInterpolation: {}".format(scale[0], scale[1],
                ↪  interpolations[interpolation]),
                "scaling_scale_{}_{}".format(scale, interpolations[interpolation]),
```

```
            output_dir)

    scales = [(0.5,0.5), (1.5,0.5)]


grid = create_grid(h, w)
scale = [1.2, 1.2]
theta = 25
translate = [5, 5]
S = scaling_matrix(scale[0], scale[1])
R = rotation_matrix(theta)
T = translation_matrix(translate[0], translate[1])
A = create_affine_matrix(O, [T, R, S])
transformed_grid = np.round(A  @ grid).astype(np.int)
transformed_image = get_transformed_image(img, grid, transformed_grid, 'linear')
plotit(transformed_image,
    "$s_x$=$s_y$={}, \\theta={}, $t_x$=$t_y$={}".format(scale[0], theta, translate[0]),
    "scale_{}_{}_rotate_{}_translate_{}_{}".format(scale[0], scale[1], theta, translate[0],
    ↪  translate[1], 'Bilinear'),
    output_dir)
```