STATISTICAL AND MACHINE LEARNING APPROACHES
FOR SEMANTIC SEGMENTATION AND SURVIVAL ANALYSIS

by

Dan M. Popescu

A dissertation submitted to The Johns Hopkins University in conformity
with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
June, 2021

© 2021 D.M. Popescu
All rights reserved
Abstract

Machine learning and, in particular, deep learning have been sweeping many disciplines in recent years. Advancements in neural networks, the primary tool of deep learning, have made them the go-to approach in a variety of fields, ranging from computer vision to natural language processing. While democratizing deep learning, high-level software packages offering “black-box” neural network models can also potentially exacerbate the divide between machine learning and classical statistical learning approaches. In this manuscript, we aim to combine machine learning and statistical techniques synergistically in a way that uses the remarkable feature extraction capabilities of deep learning, without compromising statistical rigor. We apply the techniques developed while tackling two important problems in the field of Computational Cardiology:

(a) We propose a novel deep learning solution for contrast-enhanced cardiac magnetic resonance (CMR) image analysis, which produces anatomically-accurate myocardium and scar/fibrosis segmentations and uses these to calculate clinical features. Visualizing scarring and fibrosis in the heart on CMR imaging with contrast enhancement (LGE) is paramount in characterizing disease progression and quantifying pathophysiological substrates of arrhythmias. However, segmentation and scar/fibrosis quantification from LGE-CMR is an intensive manual process prone to large inter-observer variability. We trained and tested deep neural networks which apply a 3-stage approach that identified the left ventricle (LV) region of interest, segmented the ROI into viable myocardium and regions of enhancement, and, lastly, used a post-processing network to ensure that segmentations results conformed to anatomical constraints. The segmentations
were used to directly compute clinical features, such as LV volume and scar burden. LV and scar segmentations predicted by our model achieved 96% and 75% balanced accuracy, respectively, when compared to trained expert segmentations. The difference in mean scar burden between ground truth and predicted segmentations was 2%. All resulting segmentations passed morphological checks to ensure LV anatomical accuracy. We developed and validated a 3-stage deep neural network for automatic, anatomically accurate expert-level LGE-CMR segmentation, which was used for direct extraction of important clinical measures. As our model has been trained on scans from heterogeneous cohorts, it has the potential to be extended to multiple imaging modalities and patient pathologies.

(b) We develop a new deep learning technology, that directly uses LGE-CMR images to construct a parametric model in survival analysis. Sudden cardiac death from ventricular arrhythmia continues to be a major cause of mortality worldwide and a vast public health and economic burden. Current approaches to arrhythmic death risk prediction represent broad guidelines and fail to incorporate personalized, complex, large-scale clinical data and individualized phenotyping. Deep learning (DL) approaches are ideal for such data, however, most of the DL work related to arrhythmia has focused on disease classification and detection from ECG time series data. Furthermore, although mechanistically arrhythmia results from the heterogeneous scar distribution in the heart, DL on raw imaging scans, which visualize this distribution, has not been explored for risk analysis. In this work we develop a novel DL approach which blends neural networks and survival analysis to predict patient-specific survival curves from raw contrast-enhanced cardiac magnetic resonance images and clinical covariates for patients with ischemic cardiomyopathy. The DL-predicted survival curves offer accurate arrhythmic sudden death predictions at all times up to 10 years and allow for estimation of patient-specific uncertainty in predictions. The performance of this learning architecture was evaluated on multi-center internal validation data, and tested
on an external, independent test set. It achieves both high risk discrimination (con-
cordance index of 0.83 and 0.74, respectively) and high calibration (10-year integrated
Brier score of 0.12 and 0.14, respectively). We additionally demonstrate that our DL
approach learning from only raw, unsegmented cardiac images outperforms standard
survival models constructed using both non-imaging and imaging clinical covariates.
Brought to clinical practice, this technology has the potential to transform clinical
decision-making by offering accurate, generalizable, and interpretable predictions of
patient-specific survival probabilities of arrhythmic death over time.

**Thesis Readers**

Dr. Mauro Maggioni (Primary Advisor)
Professor
Department of Applied Mathematics and Statistics at
Johns Hopkins Whiting School of Engineering
Department of Mathematics at
Johns Hopkins Krieger School of Arts and Sciences

Dr. Natalia A. Trayanova
Professor
Department of Biomedical Engineering
Johns Hopkins Whiting School of Engineering and
Division of Cardiology, Department of Medicine at
Johns Hopkins School of Medicine
Dedicated to Bernadette, Louis, and Kimberly.

Your unwavering support made this possible.
Acknowledgements

I would like to express my sincerest gratitude to Dr. Natalia A. Trayanova for believing in me and giving me the opportunity be part of her lab, where I felt a real sense of freedom in my research and where I got to work on some of the most important problems in Computational Cardiology. I would like to also thank her for introducing me to my AMS advisor, Mauro Maggioni. Mauro, thank you so much not only for the math you’ve taught me, but for the way you showed me how to think and how to discover new problems. Our weekly catch-ups have been the highlight of most of my weeks at Hopkins.

I would like to also thank the many great professors I met at Hopkins who have provided terrific guidance. Dr. Dan Naiman, thank you for the relentless support and advice you have provided. Dr. Donniel Fishkind, thank you for teaching me to use Gurobi for all my scheduling needs. Dr. Jim Fill, thank you for making measure theoretic probability bearable. Dr. Sean Sun, thank you for dreaming of the space elevator with me. Dr. Carey Priebe, thank you for your advice and care you put in teaching your students. Also, thank you to Dr. Yannis Kevrekidis and Dr. René Vidal for being part of my GBO and Defense Committees and for providing valuable feedback.

My experience at Hopkins would have been much bleaker without wonderful colleagues both inside and outside the department. Jacob and Zach, thank you for being awesome friends and for routinely nerding out with me. Trayanova Lab, thank you all for putting up with my loud mouth. Special thanks to Dr. Rheeda Ali for
bearing the brunt of that.

To my undergraduate advisor, Dr. Ovidiu Lipan, thank you for all your support over the last decade. Our many conversations have truly helped shape my path. Thank you for encouraging me to pursue the Ph.D. and thank you for teaching me to love the beauty of science and mathematics.

Many thanks also go to my mom and dad, Marcela and Stelian, to my brother, as well as to my in-laws, Mary Lou and Ken. Your love and support kept me going. Lastly, I would like to thank my wife, Kim, who has constantly helped me make the best decisions.
Contents

Abstract ......................................................... ii

Dedication ....................................................... v

Acknowledgements ................................................ vi

Contents ........................................................ viii

List of Tables .................................................... xii

List of Figures .................................................... xiii

Chapter 1 Introduction ........................................ 1
  Contribution .................................................... 2
    Neural Network Constrained Semantic Segmentation .......... 2
    Deep Neural Network Survival Analysis ....................... 4
  Outline of the Thesis ........................................ 5

Chapter 2 Preliminaries and Background .................... 6
  General Principles of Machine and Statistical Learning ...... 6
  Classification by Learning Type ................................ 8
  Classification by Task Type .................................... 9
  Training and Validation ....................................... 11
  Artificial Neural Networks ................................... 12
Chapter 3  Anatomically-Informed Deep Learning on Contrast-Enhanced Cardiac MRI for Scar Segmentation and Clinical Feature Extraction

Extraction ................................. 51

Introduction ............................. 51
Methods ................................. 53
Imaging Data and Processing .............. 53
Model .................................. 55
Stage 1: Region of Interest Segmentation Network .......... 55
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Deep Learning Survival Analysis For Arrhythmia Prediction</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Methods</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Survival Model</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Statistical Fit</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Performance Metrics</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>Neural Network Architecture</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>Training and Testing</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Patient Population and Data Sets</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>LGE-CMR Acquisition</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>Clinical Data and Primary Endpoint</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>Data Preparation</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>Statistical Analysis</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>Results</td>
<td>89</td>
</tr>
</tbody>
</table>
List of Tables

3-I   ACSNet LGE-CMR Segmentation Performance  ..........  66
3-II  ACSNet LGE-CMR Clinical Feature Performance  ..........  68
3-III Comparison of LGE-CMR Segmentation Results  ..........  69

4-I   Hyperparameter Tuning  .................................  80
4-II  Clinical Covariate Data  .................................  88
4-III Model Comparison  .................................  94
4-IV SSCAR Detailed Performance  .................................  98
List of Figures

Figure 1-1 Late-Gadolinium Enhanced Cardiac Magnetic Resonance Image 3

Figure 2-1 Machine Learning Workflow 7
Figure 2-2 Classification of machine learning algorithms by task type 10
Figure 2-3 Two-layer Neural Network 15
Figure 2-4 Backpropagation Algorithm 24
Figure 2-5 Convolutional Neural Network 29

Figure 3-1 Conversion Process to “LGE-like” Cine Images 54
Figure 3-2 ACSNet Overview 56
Figure 3-3 Segmentation Network Architecture 57
Figure 3-4 Post-processing Autoencoder Network Architecture 59
Figure 3-5 Autoencoder Latent Space Fit 61
Figure 3-6 Segmentation Results by Region 65
Figure 3-7 Scar Segmentation Results 67
Figure 3-8 Scar and LV Volume Calculation Results 70

Figure 4-1 Schematic Overview of SSCAR 81
Figure 4-2 SSCAR Interpretation 84
Figure 4-3 SSCAR Overall Performance 92
Figure 4-4 SSCAR Results at Various Time Points 93
Figure 4-5 Individual Patient Survival Probability 95
Chapter 1

Introduction

Machine learning is quickly becoming an integral part of many fields, including image processing, speech recognition, online advertising, medicine, etc. Arising as a paradigm shift in the way humans approached artificial intelligence, machine learning abandoned the classical artificial intelligence concept of requiring certain rules (the program) and data in order to generate answers; instead, programmers no longer have to craft rules \textit{a priori}, but provide the algorithm with data inputs and outputs and let the machine generate the rules [1]. The rapid growth of machine learning is in great part due to the availability of powerful hardware and large datasets, that began in the 1990, has solidified machine learning’s place as one of the most popular subfields of AI. Further developments in graphics processing units coupled with implementation of neural network algorithms in CUDA have made deep learning — a branch of machine learning which relies on subsequent transformations of the data (layers) in increasingly meaningful representations — the state-of-the-art approach for many artificial intelligence tasks [2].

The wide availability of open-source machine learning software, such as sklearn [3] and Tensorflow [4], has made it very easy for researchers in many fields to adopt these techniques in their work. For example, many of heralded deep learning algorithms as the arrival of the artificial intelligence age in personalized healthcare[5, 6, 7, 8, 9]. Such exuberance has led some to be skeptical of the prospects of machine learning,
worrying of the cognitive bias of seeing all problems as nails to be tackled with the machine learning hammer [1]. This uneasiness is also oftentimes exacerbated by the sense that machine learning algorithms act like “black boxes” [10].

The balance between model explainability and performance is crucial to practitioners, but one on which that data scientists tend to focus oftentimes later in the development process. This was particularly true in the case of deep neural networks in image recognition tasks. As neural networks become more prevalent in medicine, the desire to have explainable models has been rekindled. This is especially true in cases where data is scarce and achieving statistical confidence in results becomes onerous.

In this thesis, the goal is to re-establish the close relationship between statistical and machine learning [11] by developing deep neural network-based models backed by statistical theory. This has the potential to not only provide the regularization needed for neural networks to perform well on small data sets, but also elevate standard statistical model by equipping them with enhanced feature selection. Specifically, it will be shown how Gaussian mixture models can be used to model latent encodings of a convolutional autoencoder and used to augment the latent space. In turn, this acts as a data augmentation technique to a segmentation tasks which incorporate highly complex geometrical constraints. Additionally, traditional survival models will be re-visited and enhanced to take advantage of deep learning feature selection using fully connected and convolutional neural networks.

**Contribution**

**Neural Network Constrained Semantic Segmentation**

One of the most important applications of semantic segmentation originates in the medical field. Identifying and quantifying anatomical structures is the first step is diagnosing and treating patients. Unfortunately, medical images are notoriously
difficult to segment automatically due to their scarce number, poor image quality and imprecision in ground truth annotations. On the other hand, semi-automated methods require laborious manual intervention and user expertise. A data-efficient, automated algorithm for anatomically-accurate segmentation has so far been an unmet clinical need. Here we fulfill this need by developing an anatomically-informed deep-learning (DL) approach to semantic segmentation and clinical feature extraction. We term this technology Anatomical Convolutional Segmentation Network, ACSNet. The technology enables clinical use by ensuring anatomical accuracy and complete automation. By eliminating manual interventions, this approach holds the potential to standardize medical image analysis as part of the clinical workflow. Furthermore, ACSNet’s approach for clinical feature extraction, which satisfies highly complex geometric constraints without stunting the learning process, has the potential for broad applicability in computer vision beyond the field of medicine [12].

To demonstrate the performance of the model, we apply it to a data set of late gadolinium enhanced (LGE) cardiac magnetic resonance (CMR) images (see 1-1. Many heart pathologies are assessed by quantifying the scar tissue visible uniquely in
LGE-CMR imaging. LGE-CMR is a notoriously finicky imaging modality. Timing post-contrast administration determines the intensity difference between blood pool and scar, both of which are bright in LGE-CMR. Delineating the endocardium proves challenging for humans and computer programs alike if the blood is too similar in brightness to the fibrotic tissue. Additionally, viable tissue is dark despite being part of the same myocardium as the fibrosis. Therefore, high inter-observer variability among segmentations is unsurprising.

**Deep Neural Network Survival Analysis**

Over the last several decades, sudden cardiac death (SCD) has been a leading cause of mortality worldwide, with rates of 15–20% in Western societies and an incidence of 50 to 100 per 100,000 in the general population in Europe and North America [13, 14]. A majority of all SCD (e.g., 70% in Western countries) have coronary heart disease as the substrate, resulting in SCD from arrhythmia (SCDA)[15]. Despite the prospects of mitigating ventricular arrhythmias using prophylactic insertion of implantable cardioverter devices (ICD), current clinical guidelines for arrhythmia risk stratification — i.e., left ventricular ejection fraction (LVEF) < 30–35% [16] — only capture a mere 20% all SCDA[17], highlighting the critical need to develop personalized, accurate, and cost-effective arrhythmia risk assessment tools and alleviate this vast public health and economic burden. Although there have been many attempts to go beyond LVEF and extract more predictive features from clinical images, past approaches have fallen short by not allowing the imaging data itself to drive the feature extraction process, resulting instead in researcher-designed features that are too coarse to capture nuances in imaging data, highly mathematical features with little physiological underpinning, or complex, exceedingly computationally intensive Physics-based models. Furthermore, in previous work, SCDA risk prediction focused on single fixed time points and lacked proper integration of imaging features with broader, non-imaging clinical data to
create an overall survival model.

In the current work, we present a deep neural arrhythmia prediction network, termed SSCAR, which uses a 3-D convolutional sub-network directly on cardiac magnetic resonance images to discover and use features able of predicting individual patient times to SCDA ($T_{SCDA}$). Clinical covariate data (e.g., demographics, medications, etc.) are incorporated using a second fully-connected sub-network seamlessly integrated within an overall survival model. The main strength responsible for SSCAR’s performance comes from using the deep learning (DL) algorithm to systematically transform the CMR and clinical covariate data to learn the best fit of the available survival data, instead of a priori selecting, often arbitrary, features. Using a statistical model to describe $T_{SCDA}$ by equipping the neural network the appropriate likelihood-based objective function leads to calibrated survival probability curves for up to 10 years after the CMR scan. Directly predicting both the most probable $T_{SCDA}$ for each patient, as well as the uncertainty around it, has the potential to significantly shape clinical decision-making, offering clinicians more than simple “at risk/not at risk” prediction, but a sense of “how sure” the model is about each prediction.

**Outline of the Thesis**

The rest of the thesis will explore the two aforementioned applications in mathematical detail. First, the mathematical foundations, as well as current relevant literature, will be reviewed in Chapter 2. This chapter will go into detail on the foundations of neural networks, specific types of layers, as well as review basic concepts in Statistics pertaining to maximum likelihood estimation, Gaussian mixture models, and survival analysis. In Chapter 3, we present a novel neural network approach to semantic segmentation in medical imaging data. In Chapter 4, we develop a deep learning approach to survival analysis. Finally, Chapter 4 will provide some concluding remarks and potential future directions for this work.
Chapter 2

Preliminaries and Background

General Principles of Machine and Statistical Learning

Parts of this section are adapted from Trayanova et al. [1].

The concept of machine learning arose as a paradigm shift in the way humans approached artificial intelligence. Traditional artificial intelligence required certain input rules (the program) and data to be executed by the program to generate answers. Machine learning turned this concept on its head by eliminating the need for programmers to craft rules and instead using data and extracting answers from the data to construct the rules. The boom of machine learning, in great part due to the availability of powerful hardware and large datasets, that began in the 1990s [2], has solidified machine learning’s place as one of the most popular subfields of artificial intelligence. Although closely related to several other fields — such as mathematical statistics, probability, information theory, decision theory — machine learning tackles problems often involving extremely large and complex data (e.g., millions of images consisting of several thousand pixels), for which traditional statistical techniques may be impractical. Consequently, with the mathematical and theoretical underpinnings of the field still in a nascent state, machine learning has become a more engineering-oriented approach.
Researchers use their experience to condense raw data into meaningful representations. ML models use the training data to systematically extract features.

Figure 2-1. Typical workflow of the machine learning approach. After the data gathering step, data are split into a train set and a test set. Features (useful representations of the data) are then extracted from the training data, either by performing researcher-defined transformations of the data (feature engineering) or using machine learning techniques (feature learning). Depending on the availability of targets (expected answers from the data) and the desired machine learning task, features can be used in either a supervised or unsupervised setting. In the supervised setting, a model is trained by iteratively minimizing a loss function, which adjusts the model’s parameters such that predictions and targets match. The resulting best model is then used on the test data. In the unsupervised setting in which there are no targets available, data can be used for visualization or identifying subgroups with common characteristics, that is, clusters.
A machine learning system needs to be explicitly trained to design rules. Once the system is trained, it proceeds to the testing phase in which its ability to generalize — predict answers for never-before-seen data — is evaluated (see Fig. 2-1). This ability to categorize or predict based on data not used in the training phase is the central tenet in machine learning. To facilitate the training phase and increase the machine learning system’s (i.e., the model’s) ability to generalize, preprocessing steps are oftentimes needed[18]. For example, in a task of predicting whether given images represent cats or dogs, such preprocessing steps may aim to eliminate the variability among data by cropping all the images to the same size or rescaling the pixel intensities to a fixed range. Applying the same preprocessing methods, called feature extraction, developed on the training set, to the test set improves the machine learning model generalizability, facilitating classification of new images. Traditionally, the main approach to feature extraction has been feature engineering, in which the researcher uses their experience to condense information from the raw data into values to be used in machine learning. For example, instead of using raw ECG data as a covariate in a disease prediction machine learning problem, the researcher may only use the QRS duration. More recently, feature learning has taken the role of feature engineering, shifting the task of uncovering features to the machine learning model itself by systematically exploring the available training data.

**Classification by Learning Type**

Machine learning tasks are typically dichotomized based on whether expected answers (also known as targets or labels) are available for the training input data. For example, in a risk prediction task, the inputs would be patient characteristics and the targets would be the clinical outcome of whether the patient had/did not have an arrhythmia. If the case where targets are available for the input data and the goal is to predict targets for the unseen data, the learning task is called supervised. For example, most
classification tasks would fall under this category, where the machine learning system’s
goal is to learn features of the class (e.g., green eyes in pictures of cats and dogs are
indicative of the class cat) and apply them when new data is presented. However, if
target data is not available, the task is called unsupervised learning. The aim of such
tasks is to find useful representations of the data via various transformations such
that data can be better visualized by humans, thereby revealing simpler or reduced
structures. Although beyond the scope of this review, there are also hybrid approaches.
One such approach is semisupervised learning, in which labeled training data are
only partially available, but the presence of nonlabeled data can better guide the
supervised task. Finally, reinforcement learning [19] is a type of machine learning in
which programs go through a process of trial and error to pick actions that maximize
rewards. Unlike supervised learning, there is no need to provide labels for this machine
learning learning type, as the system uses its environment and the reward function
to guide itself. A particularly noteworthy application of this method is DeepMind’s
Alpha Go [20] algorithm, which gained such a deep understanding of the game Go
that it would set up counter-intuitive (for humans) strategies in the short term, only
to capitalize on them tens of moves later.

**Classification by Task Type**

Attempting a comprehensive taxonomy of machine learning algorithms is beyond the
scope of this dissertation and is bound to do disservice to many algorithms that have
been adapted for multiple tasks. Instead, we discuss 4 of the most important types of
tasks that machine learning tackles (see Fig. 2-2): classification, regression, clustering,
and dimensionality reduction. In the next section, we will focus on two examples.
Classification tasks typically fall under the supervised learning category and deal with
the assigning each input data point to one of a finite number of discrete categories (e.g.,
“given a raw ECG signal, is it indicative of a diseased heart?”). Regression tasks are
also supervised and expand on the classification problem by predicting $\geq 1$ continuous variables (e.g., “Given a stack of heart MRIs, what is the predicted heart muscle volume?”). The forecasting role that regressions play in ML should be contrasted with that of regressions in applied statistics, which is to establish causal relationships between dependent and independent variables.

![Classification of machine learning algorithms by task type](image)

**Figure 2-2.** Classification of machine learning algorithms by task type. UpSet plot [21] showing algorithms (columns) that can be used for a given task type (rows: regression, classification, dimensionality reduction, and clustering) using black filled-in circles. Gray circles denote algorithms not typically used for the respective task. Connected circles denote algorithms that are used for multiple tasks. Algorithms in blue rectangles are typically supervised, those in red ellipses are typically unsupervised, and algorithms in red and blue can be either supervised or unsupervised. ANN indicates artificial neural network; Clst, clustering; GMM, Gaussian mixture model; Diff, diffusion; Dim, dimensionality; K-NN, k-nearest neighbors; LDA, linear discriminant analysis; LLE, local linear embedding; LS, least squares regression; PCA, principal component analysis; RF, random forest; SVM, support vector machines; SVR, support vector regression; and tSNE, t-distributed stochastic neighbor embedding.

In certain machine learning problems, input training data is available without
associated labels. The goal here may be identify subgroups of the data which display some degree of similarity, that is, clustering. For example, used on a diverse set of patients, clustering may help identify certain subcohorts that help clinicians better understand the population. Additionally, one may be interested in reducing the dimension of the data for various purposes, including visualization in 2 or 3 dimensions, or as part of a feature extraction task.

Training and Validation

For most of the algorithms discussed, the process of training, particularly in the case of supervised learning, is based on defining a loss function that will be minimized, typically through an iterative process (e.g., the loss for a simple linear regression is the mean squared error). Some algorithms may have additional degrees of freedom (e.g., number of layers in an ANN), which are chosen by the researcher. These model design parameters — as opposed to parameters resulting from the loss minimization algorithm — are called hyperparameters. Before performing any training or optimization, a test set should be identified and sequestered from the training data. Depending on the problem and amount of available data, this could be a percentage (10%–30%) of the training data or an entirely separate external validation cohort from a different source.

In practical applications, the goal is to use available training data and select a model (algorithm and hyperparameters) that will have the best predictive performance on new data [18]. The performance may be assessed using the loss function or other intuitive metrics chosen by the researcher — for example, true/false positive/negatives and other measures derived from them. If the training data is plentiful, one easy solution is to carve out a percentage of the training data and use it as a validation set. This will serve as an estimate for the model’s performance on the test set and is often used to optimize hyperparameters of the algorithm by choosing the hyperparameters which maximize performance on the validation set. Unfortunately, there are no rigid
rules on what constitutes enough data, as this can depend on the difficulty of the
learning task, the algorithm used, the desired statistical confidence, etc.. In real
applications, however, there is usually a scarcity of training and testing data, and
more rigorous validation techniques are needed. One staple technique is $k$-fold cross-
validation [22]. This requires the researcher to split the training data in $k$ (typically, 5
or 10) subsets and repeatedly set one subset for validation and train on the remaining
$k - 1$. The performance for this split of the data will be the average across the $k$
different folds, and this process can be further repeated with other random splits,
followed by an average across all the runs. One of the downfalls of this approach is
that, despite giving more stable performance estimates, the number of times the model
needs to be trained can be computationally expensive. Another popular validation
method is the bootstrap [23]. The main idea behind this method is to sample the
training set with replacement, train the model on this bootstrapped set (which may
contain duplicates), and estimate the generalization error as the discrepancy between
performance on the bootstrapped set compared to the whole set, averaged over many
repetitions of this process. Although various estimates have been proposed for the
bootstrap generalization error, one of the main drawbacks remains that the estimates
tend to be overly optimistic, especially in models that are prone to overfit. It is
important to note there are no universal validation methods and analysis of these
methods remains a very hard problem [24].

Artificial Neural Networks

Although the term “neural network” has its origins in biology, where they were posited
as mathematical representation of biological systems [25], artificial neural networks
gained significantly more recognition in the world of pattern recognition. The first
part of this section will contain background on theoretical work, in which case the
attention will be restricted to simple neural networks called multilayer perceptrons
(or fully connected, or dense). In the second part, more attention will be given to the practical matters of neural network models, such as exploring different layers and architectures, the backpropagation algorithm, optimizer choices, and loss functions.

**Definition**

The majority of the this section is based on Devroye, Györfi and Lugosi [24], with some details omitted for brevity.

We consider the binary classification problem with classes 0 and 1 and let $\varphi(x)$ be a discriminant, together with the decision

$$
\phi(x) = \begin{cases} 
0 & \text{if } \varphi(x) \leq 1/2 \\
1 & \text{otherwise}.
\end{cases}
$$

where we let $\varphi$ be a neural network without hidden layers:

$$
\varphi(x) = c_0 + c^T x,
$$

where the $c_i$'s are weights, $x = (x^{(1)}, \ldots, x^{(d)})^T$, and $c = (c^{(1)}, \ldots, c^{(d)})^T$. For a feed-forward neural network, with one hidden layer, one takes

$$
\varphi(x) = c_0 + \sum_{i=1}^k c_i \sigma(\varphi_i(x)),
$$

where the $c_i$'s are weights as before, $\varphi_i$'s take the form in (2.1): $\varphi_i(x) = b_i + \sum_{j=1}^d a_{ij} x^{(j)}$, for some constants $b_i$ (called bias) and $a_{ij}$ (called kernel). The number $k$ denotes the number of “hidden” neurons in this layer. The function $\sigma(x)$, also called a sigmoid or activation function. Historically, these were sigmoid-shaped functions, that is $\sigma(x) \to -1$ and $\sigma(x) \to 1$ as $x \uparrow +\infty$, but functions easier to compute have become popular in practice. Some common examples include:

- the threshold sigmoid

$$
\sigma(x) = \begin{cases} 
-1 & \text{if } x \leq 0 \\
1 & \text{if } x > 0
\end{cases}
$$
• the standard or logistic sigmoid

\[ \sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \]

• the hyperbolic tangent

\[ \sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]

• the rectified linear unit (ReLU)

\[ \sigma(x) = \max\{0, x\} \]

• the softplus

\[ \sigma(x) = \ln (1 + e^x) \]

The process described above can be continued to form a multi-layer feed-forward neural network. For example, a two-hidden-layer neural network would take the form

\[ \varphi(x) = c_0 + \sum_{i=1}^{l} c_i z_i, \]

where

\[ z_i = \sigma \left( d_{i0} + \sum_{j=1}^{k} d_{ij} u_j \right), \quad 1 \leq i \leq l, \]

and

\[ u_j = \sigma \left( b_j + \sum_{i=1}^{d} a_{ji} x^{(i)} \right), \quad 1 \leq j \leq k, \]

and \( d_{ij}'s, b_j's, \) and \( a_{ji}'s \) are constants. The first hidden layer has \( k \) neurons and the second has \( l \) (see Fig. 2-3).

**Theoretical Considerations**

If the number of hidden neurons \( k \) is allowed to grow unboundedly with the size of the data \( n \), it turns out that one can prove universal consistency for the one-hidden-layer neural network. It is, therefore, of little theoretical interest to consider network
Figure 2-3. Schematic representation of a two-hidden-layer feed-forward neural network. The input $x$ is sequentially transformed by a linear combination with coefficients $a_{ij}$ (kernel and bias), after which a sigmoid activation function is applied. The process is repeated for each hidden layer. Finally, the output is passed through a perceptron which outputs the predicted class 0 or 1.

Architectures more complicated than the class $C^{(k)}$ given by Eq. (2.2). However, there may be information-theoretic gain of considering more layers, where similar performance can be obtained with a significantly reduced number of hidden neurons. At least for classification, little will be gained by going beyond two layers. Incidentally, modern neural network architectures in deep learning have hundreds of layers and can achieve excellent performance in real-world applications. We will discuss such considerations in a later section.

We first establish some notation and basic results regarding classification. Assume we are given data $(X, Y)$ in $\mathbb{R}^d \times \{1, \ldots, M\}$ for the $M$-class classification problem. Let $g$ be a classifier which is one’s guess of $y$ given $x$. Naturally, an error occurs if $g(X) \neq Y$. We write for the probability of error of a classifier $g$

$$L(g) = \mathbb{P}\{g(X) \neq Y\}.$$
There exists a best possible classifier $g^*$, which is defined by

$$g^* = \arg \min P\{g(X) \neq Y\}.$$ 

Obviously, $g^*$ will depend on the (typically unknown) distribution of the data $(X, Y)$. The problem of finding $g^*$ is called the Bayes problem and the classifier $g^*$ is called the Bayes classifier (or Bayes rule). The minimal probability of error, also called the Bayes error and is denoted by $L^* = L(g^*)$. In practical application, the distribution of the data is unknown and, therefore, so is $g^*$. However, most theoretical results use $L^*$ as the “holy grail” to which empirical results aspire. We begin with an important result regarding plug-in decisions. The best guess $Y$ from the observation $X$ is the Bayes decision

$$g^*(x) = \begin{cases} 0 \text{ if } \eta(x) \leq 1/2 \\ 1 \text{ otherwise} \end{cases} = \begin{cases} 0 \text{ if } \eta(x) \leq 1 - \eta(x) \\ 1 \text{ otherwise} \end{cases}.$$

The function $\eta$ is typically unknown. However, we may have access to a nonnegative function $\tilde{\eta}$ which approximates $\eta$, in which case it is natural to consider the plug-in decision function

$$g(x) = \begin{cases} 0 \text{ if } \tilde{\eta}(x) \leq 1/2 \\ 1 \text{ otherwise} \end{cases}$$

to approximate the Bayes decision. To make this concrete, we have the following theorem (see [24] for the proof)

**Theorem 2.1.** For the probability error of the plug-in decision $g$ defined above, we have

$$P\{g(X) \neq Y\} - L^* = 2 \int_{\mathbb{R}^d} |\eta(x) - 1/2| 1_{\{g(x) \neq g^*(x)\}} \mu(dx)$$

and

$$P\{g(X) \neq Y\} - L^* \leq 2 \int_{\mathbb{R}^d} |\eta(x) - \tilde{\eta}(x)| \mu(dx) = 2E|\eta(x) - \tilde{\eta}(x)|.$$

We now consider the class $C^{(k)}$ of classifiers (2.2) consisting of all neural networks with the threshold sigmoid and $k$ hidden nodes in two hidden layers. Let $D_n$ be the
training data, with the goal of selecting a classifier from $C^{(k)}$. For good classifiers, it is necessary that the best rule in $C^{(k)}$ has the probability of error close to $L^*$, that is

$$\inf_{\phi \in C^{(k)}} L(\phi) - L^*$$

is small. This approximation error will naturally be positive for most distributions for fixed $k$ and go down for large $k$. The question of interest is whether this holds for all distributions $(X, Y)$. Using an argument based on arrangements, it is show in [24] that this is in fact true as given by the following theorem.

**Theorem 2.2.** If $C^{(k)}$ is the class of all neural network classifiers with the threshold sigmoid and $k$ neurons in two hidden layers, then

$$\lim_{k \to \infty} \inf_{\phi \in C^{(k)}} L(\phi) - L^* = 0$$

for all distributions of $(X, Y)$.

It turns out that this property holds for more general sigmoid functions as well. We let $\varphi$ as in (2.2) and using Theorem (2.1), we have

$$L(\phi) - L^* \leq 2\mathbb{E}\{|\varphi(X) - \eta(X)|\},$$

where $\eta(x) = \mathbb{P}\{Y = 1|X = x\}$. Thus, $\inf_{\phi \in C^{(k)}} L(\phi) - L^* \to 0$ as $k \to \infty$ if

$$\mathbb{E}\{|\varphi_k(X) - \eta(X)|\} \to 0$$

for some sequence $\{\varphi_k\}$ with $\phi_k \in C^{(k)}$ for $\phi_k(x) = 1_{\{\varphi_k(x) > 1/2\}}$. To have universal consistency, we need only ensure that the family of $\varphi$’s can approximate any $\eta$ in the $L_1(\mu)$ sense. That is, the approximation error $\inf_{\phi \in C^{(k)}} L(\phi) - L^*$ converges to zero if the class of functions $\varphi$ is dense in $L_1(\mu)$ for every $\mu$. Another condition— much too restrictive — sufficient for this is that the the class $\mathcal{F}$ of functions $\varphi$ becomes dense in the $L_\infty$ norm in the space of continuous functions $C[a, b]^d$, where $[a, b]^d$ is a hyperrectangle in $\mathbb{R}^d$, for any $a$ and $b$. The following results tackle precisely this.
Lemma 2.3. Assume that a sequence of classes of functions $F_k$ become dense in the $L_\infty$ norm in the space of continuous functions $C[a, b]^d$ (where $[a, b]^d$ is the hyperrectangle of $\mathbb{R}^d$ defined by $a$, $b$). In other words, assume that for every $a, b \in \mathbb{R}^d$, and every bounded function $g$,

$$ \lim_{k \to \infty} \inf_{f \in F_k} \sup_{x \in [a, b]^d} |f(x) - g(x)| = 0. $$

Then for any distribution $(X, Y)$,

$$ \lim_{k \to \infty} \inf_{\phi \in C^{(k)}} L(\phi) - L^* = 0, $$

where $C^{(k)}$ is the class of classifiers $\phi(x) = 1_{\{\phi > 1/2\}}$ for $\phi \in F_k$.

Proof. The proof follows easily from Theorem 2.1. \qed

Using the denseness of the class of trigonometric polynomials in the $L_\infty$ sense for $C[0, 1]^d$ — a special case of the Stone-Weierstrass theorem —

Theorem 2.4. For every continuous function $f : [a, b]^d \to \mathbb{R}$ and for every $\epsilon > 0$, there exists a neural network with one hidden layer and function $\phi(x)$ as in (2.2) such that

$$ \sup_{x \in [a, b]^d} |f(x) - \phi(x)| < \epsilon $$

However, the convergence in the previous theorem may be arbitrarily slow for some $f$. With a restricted class of functions, one can obtain upper bounds on the rate of convergence [26]. Using Theorem 2.4 and Lemma 2.3, one can prove the corollary:

Corollary 2.4.1. Let $C^{(k)}$ contain all neural network classifiers defined by networks of one hidden layer with $k$ hidden nodes, and an arbitrary sigmoid $\sigma$. Then for any distribution $(X, Y)$,

$$ \lim_{k \to \infty} \inf_{\phi \in C^{(k)}} L(\phi) - L^* = 0, $$

The above convergence also holds if the range of parameters $a_{ij}$, $b_i$, $c_i$ is restricted on the interval $[-\beta_k, \beta_k]$, where $\lim_{k \to \infty} \beta_k = \infty$. 18
While the result above highlights a very desirable property of simple neural networks, namely that as the architecture (here, $k$) is allowed to grow, the approximation error vanishes, this fact is mostly of theoretical interest. In practice, the architecture is given and the task is to adjust the weights of the network depending on the data $D_n$. In this case, it is interesting to analyze the probability of error of the chosen rule compared to $\inf_{\phi \in C^{(k)}} L(\phi)$. In the following discussion, it is assumed that the reader is familiar with the Vapnik-Chervonenkis theory, VC dimension, shattering number (cf. Chapter 12 in [24]).

We are now in the setting where data $D_n = ((X_1, Y_1), \cdots (X_n, Y_n))$ is used to tune the network weights. The VC dimension $V_C^{(k)}$ of the class $C^{(k)}$ is known to determine the performance of the learning algorithm in terms of the probability of error. It is already known that for arbitrary distributions of the data, no classifier picking method can perform better than $\sqrt{V_C^{(k)}/n}$ in terms of rate. However, there exists a way of choosing the parameters of the network by minimizing the empirical error probability such that the resulting classifier $\phi_n^*$ satisfies

$$\mathbb{E}\{L(\phi_n^*)\} - \inf_{\phi \in C^{(k)}} L(\phi) \leq 16\sqrt{\frac{V_C^{(k)} \log n + 4}{2n}}$$

for all distributions. Unfortunately, in the case $V_C^{(k)} = \infty$, for any $n$ and for any rule, there exist distributions with arbitrarily poor performance. For arbitrary sigmoids $\sigma$, Baum [27] showed that neural networks with $k$ hidden nodes in one hidden layer have $V_C^{(k)} \geq 2[k/2]d$. This bound can be improved if one only considers the threshold sigmoid, leading to the following important result:

**Theorem 2.5.** Let $\sigma$ be the threshold sigmoid. Let $g_n$ be a classifier of from $C^{(k)}$ that minimizes the empirical error

$$\hat{L}_n(\phi) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{\{\phi(X_i) \neq Y_i\}}$$

over $\phi \in C^{(k)}$. If $k \to \infty$ such that $k \log n/n \to 0$ as $n \to \infty$, then $g_n$ is strongly
universally consistent, that is,

\[ \lim_{n \to \infty} L(g_n) = L^* \]

almost surely for all distributions of \((X,Y)\).

This result is very powerful as guarantees that a sequence of appropriately sized neural networks may be trained to yield the optimum probability of error in the limit. The caveat is that the sigmoid activation functions are thresholds. Relaxing the requirement on sigmoids greatly complicates the problem, as the VC dimension depends on the form of the sigmoid. Moreover, sigmoids alone can cause the VC dimension to blow up to infinity. Fortunately, it was proved that this is not the case for common activation functions, such as the logistic and arctan. Unfortunately, the result of Theorem 2.5 remains of primarily theoretical interest, as there are no efficient algorithms to find the classifier that minimizes the empirical error probability. Much more popular are networks trained to minimize empirical \(L_p\) probability losses. Although algorithms developed for this purpose may run in reasonable time, they may get stuck in local optima, resulting in classifiers with hard-to-predict probability of error. One of the most famous such algorithms is the backpropagation [28], which will be discussed shortly.

It turns out that focusing on \(L_p\)-based empirical losses has the benefit of obtaining a general consistency result for all sigmoids. The process of minimizing such losses is similar to finding a regression function, potentially increasing robustness. In fact, White [29] proved consistency of neural network estimates based on squared error minimization in the setting of regression function estimation. We state the main result and direct the reader to [24] for the proof.

**Theorem 2.6** (Lugosi and Zegner (1995)). Let \(\sigma\) be an arbitrary sigmoid. Define the class \(\mathcal{F}_n\) of neural networks by

\[
\mathcal{F}_n = \left\{ \sum_{i=1}^{k_n} c_i \sigma(a_i^T x + b_i) + c_0 : a_i \in \mathbb{R}^d, b_i \in \mathbb{R}, \sum_{i=0}^{k_n} |c_i| \leq \beta_n \right\},
\]
and let $\varphi_n$ be a function that minimizes the empirical $L_1$ error

$$J_n(\varphi) = \frac{1}{n} \sum_{i=1}^{n} |\varphi(X_i) - Y_i|$$

over $\varphi \in \mathcal{F}_n$. If $k_n$ and $\beta_n$ satisfy

$$\lim_{n \to \infty} k_n = \infty, \quad \lim_{n \to \infty} \beta_n = \infty, \quad \text{and} \quad \lim_{n \to \infty} \frac{k_n \beta_n^2 \log(k_n \beta_n)}{n} = 0,$$

then the classification rule

$$g_n(x) = \begin{cases} 0 & \text{if } \varphi_n(x) \leq 1/2 \\ 1 & \text{otherwise} \end{cases}$$

is universally consistent.

A nearly identical result is obtained for the regression problem using $L_p$-based empirical risk minimization[30].

While it is important to have good grasp on the restrictions needed to prove universal consistency for neural networks in the distribution-free setting, these restrictions are far too stringent when considering state-of-the-art neural network architectures used in modern applications. Consequently, the next section will relax the level of rigour and address practical considerations arising from implementing such networks in the real, small-data settings, where limiting results are of lesser importance.

**Practical Considerations**

Over the last decade, artificial neural network have seen a tremendous resurgence in popularity after decades of being largely ignored by the scientific community. With important contributions by Geoffrey Hinton at the University of Toronto, Yoshua Bengio at the University of Montreal, Yann LeCun at New York University, and IDSIA in Switzerland, the field of artificial neural networks began its comeback. In 2011, Dan Ciresan started winning academic image classification competitions with GPU-trained deep neural networks marking one of the first practical successes of modern deep
learning. Starting with 2012, deep convolutional neural networks became the go-to computer vision algorithm for imaging-related tasks [31]. Not surprisingly, with such quick progress in deep learning algorithms, rigorous mathematical theory on the statistical properties of these algorithms has lagged behind for the more sophisticated algorithms. In this section, we will go over some of the more applied aspects of artificial neural networks.

**Error Backpropagation**

Training an artificial neural network is the process of adjusting the model’s weights (coefficients) in order to minimize some loss function which is typically a measure of (dis)agreement between the true labels associated with the data and the neural network prediction of these labels. This process involves two stages. In the first stage, the derivatives of the loss function with respect to the weights need to be evaluated. In the second stage, the weights are adjusted based on the calculated derivatives using one of many possible optimization schemes, which we will discuss in the next section. The goal of this section is to describe backpropagation [28] — a computational efficient way of calculating the derivatives in the first stage of the training process. It should be noted that in recent years, modern neural networks are implemented using symbolic differentiation, eliminating the need of implementing backpropagation “by hand”.

For the sake of simplicity, we illustrate the backpropagation algorithm on a simple feed-forward neural network with no bias coefficients. This section follows the exposition in [11]. Assume the \( w \) be the weights of a the neural network and consider an input vector \( x_n \) to the network. Consider \( z_i \) is the activation of a unit that is getting passed down to a unit \( j \) according to:

\[
a_j = \sum_i w_{ji} z_i, \quad (2.3)
\]

where \( w_{ji} \) is the weight associated with the connection between nodes \( j \) and \( i \). the
final output of unit $j$ after applying the activation function $\sigma$ becomes:

$$z_j = \sigma(a_j). \quad (2.4)$$

Next, let $L_n$ be the contribution to the loss function of the input $n$. We omit the subscript $n$ from the network variables and write for the derivative of $L_n$ with respect to the weights using the chain rule:

$$\frac{\partial L_n}{\partial w_{ji}} = \frac{\partial L_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \quad (2.5)$$

We use the notation

$$\delta_j := \frac{\partial L_n}{\partial a_j} \quad (2.6)$$

for what we refer to as errors, for reasons which will be clear shortly. Using Eq. (2.3), we can write

$$z_i = \frac{\partial a_j}{\partial w_{ji}}. \quad (2.7)$$

Using the expression for the errors in (2.6) and (2.7) and substituting in (2.5), we then obtain

$$\frac{\partial L_n}{\partial w_{ji}} = \delta_j z_i. \quad (2.8)$$

Equation (2.8) shows exactly how the sought after derivative should be computed. To compute the derivative of the loss with respect to the weight between intermediary input $i$ and intermediary output $j$, one needs to multiply the error associated with node $j$ with the input value $z_i$ at the preceding node $i$. Since the $z$’s are computed in the forward pass through the network (see Fig. 2-4, black arrow), the only thing remaining is to compute the values of $\delta_j$ for each hidden and output unit in the network then apply (2.8) repeatedly. To find the errors, we begin by computing them at the output nodes and propagating them back to the hidden ones. The errors associated with the outputs can be computed directly from the expression of the loss function.

For example, assume there are $k$ nodes to which $j$ sends connections (Fig. 2-4) and the loss function is the mean squared error $L_n = \sum_k (y_{nk} - \hat{y}_{nk})^2$, then $\delta_k = 2(y_k - \hat{y}_k)$,
where $\hat{y}$ is the target (label). Then, the errors are backpropagated by using the fact that

$$
\delta_j := \frac{\partial L_n}{\partial a_j} = \sum_k \frac{\partial L_n}{\partial a_k} \frac{\partial a_k}{\partial a_j},
$$

where the sun runs over all units $k$ to which unit $j$ sends connections. It can be seen by using similar arguments as above that the backpropagation formula becomes

$$
\delta_j = \frac{d \sigma}{da_j} \sum_k w_{kj} \delta_k.
$$

We note that to compute the the errors at unit $j$, error information from just the $k$

downstream nodes is required in addition to the $k$ downstream weights. This process can be applied recursively moving towards the start of the network regardless of topology.

We summarize here the backpropagation algorithm procedure:

---

**Figure 2-4.** Schematic of the backpropagation algorithm showing calculation of the loss $\delta_j$ associated with the hidden node $j$ by backpropagation of the $k$ $\delta$’s associated with the downstream nodes to which $j$ sends connections. The black arrow shows the natural flow of information (forward propagation) in the network, and the red arrow indicates the backward propagation of loss information. Adapted from Bishop [11].
1. Let \( \mathbf{x}_n \) be an input vector to the neural network. Forward-propagate through the entire network using (2.3) and (2.4).

2. Evaluate the \( \delta_1, \ldots, \delta_k \) for all output units.

3. Backpropagate the \( \delta \)'s using (2.9) to find \( \delta_j \) for each hidden unit in the network.

4. Use (2.8) to evaluate the required derivatives.

Thanks to the linearity of derivatives, the total loss \( L \) can be obtained by the same steps above (even in batches) using the fact that

\[
\frac{\partial L}{\partial w_{ji}} = \sum_n \frac{\partial L_n}{\partial w_{ji}}.
\]

It is important to point out that, although throughout this section some simplifying assumptions were made, the algorithms can easily be extended to more general cases. For example, for the bias terms ignored here, we would have \( z = 1 \). Additionally, the derivation relies on a single activation function \( \sigma(\cdot) \), but this can be different for each unit.

**Weight Optimization Schemes**

With an understanding of how to compute the gradient of the loss function with respect to the weights, \( \nabla L(\mathbf{w}) \), we now turn to the problem of adjusting the weights \( \mathbf{w} \) in a way that \( L \) is minimized. There is, of course, no hope of finding stationary points by solving \( \nabla L(\mathbf{w}) = 0 \) using analytical methods. Furthermore, as Bishop [11] points out, the symmetry of the neural networks imply that for a densely-connected two-layer network with \( M \) hidden units, for any point \( \mathbf{w} \) that is a local minimum, there will be \( M!2^M \) other points in the weight space that are equivalent minima. Considering that modern neural network architectures can have number of parameters in the millions, it should be obvious that iterative numerical procedures are needed to solve the optimization problem. Having highly nonlinear dependence between the loss
and the weights, as well as potentially dealing with very large amounts of data, can seriously thwart naive approaches such as plain gradient descent.

First, the large amounts of data issue can be handled by considering mini-batch approaches. That is, use a small subset of the data drawn at random to calculate a gradient and move a small step in the corresponding direction, and then repeat. The size of the mini-batch is a trade-off between the accuracy of the gradient estimate and the cost to evaluate it. To deal with the very complicated weight space, most techniques rely on choosing some initial value \( w^{(0)} \) for the weights and applying subsequent updates of the form

\[
  w^{(\tau+1)} = w^{(\tau)} + \Delta w^{(\tau)},
\]

for some time-step \( \tau \). In most algorithms, the weight updated \( \Delta w \) depends on the gradient \( \nabla L(w) \). A complete taxonomy of optimization algorithms is beyond the scope of this review. Instead, we describe a few popular optimization algorithms which highlight two important concepts (learning rate and momentum).

**Stochastic Gradient Descent (SGD)** Probably one of the most basic and widely used algorithms for neural network optimization is SGD. The stochastic aspect was described above and has to do with only using part of the data for each update. The weight update is

\[
  \Delta w^{(\tau)} = -\rho \nabla L(w^{(\tau)}),
\]

where \( \rho \) is the learning rate. The learning rate can be a constant or can be made a function of \( \tau \) and adapt in some way. Momentum [32] can be added to the update using:

\[
  v^{(\tau+1)} = pv^{(\tau)} - \rho \nabla L(w^{(\tau)})
\]

\[
  \Delta w^{(\tau)} = v^{(\tau+1)},
\]

where \( p \) is the momentum constant and the velocity \( v \) is initialized with \( v^{(0)} = 0 \).
RMSProp  Another common and successful optimization algorithm is RMSProp, which uses a moving average of the square of the gradient to weigh the learning rate. The update is given by

\[ g^{(\tau+1)} = \gamma g^{(\tau)} + (1 - \gamma)[\nabla L(w^{(\tau)})]^2 \]
\[ \Delta w^{(\tau)} = -\frac{\rho}{\sqrt{g^{(\tau+1)}} + \epsilon} \nabla L(w^{(\tau)}) , \]

where \( \epsilon \) is a constant for stability, \( \gamma \) is the discounting factor, and the square should be read as elementwise. The idea behind RMSProp is that the newly computed gradient cannot change the direction by too much. Oftentimes, momentum is incorporated into RMSProp similarly to above.

Adam [33]  One of the algorithms that requires very little adjustment of its parameters is Adam. It is based on adaptive updates of lower order moments. The updates are given by:

\[ \Delta w^{(\tau)} = -\rho \frac{\hat{m}^{(\tau+1)}}{\sqrt{\hat{v}^{(\tau+1)}} + \epsilon} , \]

where

\[ \hat{m}^{(\tau+1)} = \frac{m^{(\tau+1)}}{1 - \beta_1^{\tau+1}} \]
\[ \hat{v}^{(\tau+1)} = \frac{v^{(\tau+1)}}{1 - \beta_2^{\tau+1}} \]
\[ m^{(\tau+1)} = \beta_1 m^{(\tau)} + (1 - \beta_1)[\nabla L(w^{(\tau)})] \]
\[ v^{(\tau+1)} = \beta_2 v^{(\tau)} + (1 - \beta_2)[\nabla L(w^{(\tau)})]^2 , \]

where the square in the gradient is taken elementwise, \( \beta_i \)'s are constant weights on the momenta \( m \) and \( v \) and \( \epsilon \) is a constant for numerical stability.

Many variants of these algorithms have been proposed and the field of optimizer schemes for neural networks remains an active area of research. There is no consensus on which algorithms work well in general, however there are some proposed general guidelines which seem to hold in practical applications.
Neural Network Layer Types

So far, we have focused on a single type of neural network hidden layer, namely, the “dense” or “fully-connected”, where each neuron has a connection to all the other neurons in the preceding and succeeding layers. While these layers are effective due to their comprehensive nature, there are many other layers which are better suited for certain data types. For example, neural network building connections between multiple time points of temporal data (also known as recurrent neural networks) use long short-term memory (LSTM) \[34\] layers, which benefit from feedback connections. Additionally, layers need not always have a state (i.e., weights). For example, as will be described in a later section, a “Dropout” layer arbitrarily sets a certain proportion of the nodes to 0 as part of a technique to combat overfitting. It is impractical and beyond the scope of this work to do a complete taxonomy of the myriad of network layer types, but special attention will be given to the convolution layers, as they are fundamental to the models described in later chapters.

Convolutional Neural Networks (CNNs). Arguably one of the successful network architectures in recent years, convolutional neural networks \[35, 36\] are specifically design to take advantage of input data in the form of images. Oftentimes tasks involving images are invariant to certain transformations (e.g., in an image classification task where the goal is identify whether an image contains a cat or a dog, rotating the image does not change the label). CNNs are designed to specifically capture this invariance and are particularly effective at dealing with imaging data. While a dense network should theoretically be able to learn these invariances, such structures don’t take advantage of the fact that in images pixels located close together tend to be highly correlated.

While it is easy to extend the discussion to higher dimensions, we only illustrate here the convolutional operation on a single-channel image. We let $I \in M^{n \times n}(\mathbb{R})$
be a square single-channel image represented by a square real-valued matrix. The weights associated with a convolutional layer are given by a kernel (or filter) \( K \in \mathcal{M}^{k \times k}(\mathbb{R}) \) with \( k < n \). The result of the convolution operator is given by the matrix \( F \in \mathcal{M}^{n-k+1 \times n-k+1}(\mathbb{R}) \), called a feature map:

\[
[F]_{i,j} = [I \ast K]_{i,j} = \sum_{l=1}^{n} \sum_{p=1}^{n} [K]_{l,p} [I]_{i+l-1,j+p-1}
\]

A schematic representation of the convolutional operation is shown in 2-5. Of note, the sliding window (receptive field) need not be done with a \textit{stride} of 1, but with larger values, resulting in a further downsampled feature map. To limit the downsampling effect, padding is often provided for the original image (\textit{i.e.}, rows and columns of 0 or any other fill value surrounding the image). If we let \( p \) be the number of rows/columns of padding and \( s > 0 \) the stride parameter, the dimension of the feature map becomes 

\[
[(n + 2p - k)/s + 1]
\]

The other key parameter in a convolutional layer besides the kernel is the number of feature maps (\textit{i.e.}, “depth”). As more convolutions are applied in sequence, the depth parameters is usually increased. To compensate, after each convolution, a downsampling layer is typically applied. The idea behind such a layer is the same, that is, to slide a small window along the resulting feature map and combine the
elements within each window in some way. For example, the max pooling layer is a standard way to downsample by only keeping the largest element within each window as it slides though the feature map. We stress again that everything discussed here can be done in tensor form for higher dimensional inputs.

**Weight regularization.**

**Regularization**

One practical problem arising in training neural networks is that of overfitting. This issue is particularly relevant when the neural networks involved have very large numbers of trainable weights (high capacity). Oftentimes the amount of available data is limited and such high capacity model will tend to “learn” the training set very rapidly, resulting in excellent performance on the training set, but very poor performance on the test set. This phenomenon is also sometimes called “lack of generalization” — that is, the network is not really learning, but memorizing the training data. To combat this problem, there are several regularization techniques which have become popular in recent years. We will describe here a few common ones, excluding the most obvious solution which is typically impossible in practice, *i.e.*, getting more training data.

**Reducing the network’s size.** If a neural network is able to quickly (*i.e.*, few gradient updates) decrease the loss function during optimization, it may be the case that the network has too many trainable weights and its capacity needs to be reduced. The process of reducing the capacity depends on the type of architecture, but typically consists of reducing the number of layers, the number of units for each layer, the number of convolution operations, increasing the stride parameter in convolutions, *etc.*. While deep learning approaches may be very effective at fitting data, the real challenge remains generalization. On the other hand, if the number of trainable
parameters is reduced too aggressively, it is possible that the network underfits and the model cannot learn useful representations. There is a clear trade-off between network capacity and generalization ability, but, unfortunately, there is no magic formula that prescribes how to tackle the capacity dilemma. Several models need to be evaluated by starting with simple architectures and adding capacity. Alternatively, systematic ways can be explored as described in the following section.

**Weight regularization.** In a very high capacity model, there should be no surprise that there are situations where not all trainable parameters are actually needed to obtain a certain fit. To coerce the neural network to look for the fit that provides the “simpler explanation”, weight regularization can be achieved by penalizing the loss function proportional to the trainable weights. In turn, this will drive some parameters to 0 in the hope of obtaining a “simpler” model. Either $L^1$, $L^2$ or a combination of the two norms can be used on the weights as a penalty.

**Layer-based regularization.** Another approach to combat overfitting is to provide the neural network with layers specifically aimed to regularize. One of the most common examples introduced by Geoff Hinton and his colleagues at University of Toronto is a Dropout layer [37]. Given a dropout rate parameter, the dropout layer will randomly set that many entries of the input tensor to 0. While it may seem counterintuitive that such a measure provides regularization, Hinton claims that “conspiracies” — happenstance insignificant patterns that make the neural network memorize the training data — can be broken down with the introduction of noise.

**Advanced Neural Network Design Principles**

The immense attention deep learning has received in the last few years has led to an abundance of ideas in the field, typically demonstrated with numerical experiments, rather than rigorous theorems. Nevertheless, we mention here a few of these ideas as
they will be used in this work.

**Batch Normalization.** Introduced by Ioffe and Szegedy [38], *batch normalization* is a very common *normalization* technique which aims to make the batches of data seen by the neural network seem similar. Effectively, a batch normalization layer will keep track of an exponentially-decaying batch-wise mean and variance and uses these to standardize each batch, essentially de-meaning and scaling to unit variance. This has been shown in practice to allow for higher network depths by handing gradient propagation in a more numerically stable way.

**Hyperparameter Optimization.** With increasing complexity of neural network architectures, the number of required hyperparameters — that is, parameters determining the network architecture, such as number of layers, number of units per layer, type of activation functions, *etc.*— increases as well. Searching through all possible hyperparameter configurations is computationally intractable, but there are techniques available which attempt to perform a meta-optimization by searching through possible hyperparameter sets. The one used in this work relies on the package *hyperopt* [39]. The algorithm takes as input a search space which defines, for each hyperparameter, the possible values the hyperparameter can take. Optionally, the universe of values for hyperparameters can be specified using a probability distribution from which values will be randomly drawn. Each model is evaluated in terms of the desired metric and an optimization algorithm informs the next parameter configuration. One typical way to perform the optimization is using the tree-structured Parzen estimator (TPE), which uses Bayes rule and non-parametric densities to draw candidate parameters in a way to maximize the expected improvement [40].

**Model ensembling.** An additional powerful technique for neural networks to get the best performance on a task is *model ensembling*. This approach consists of pooling
together multiple predictions from various models. The idea behind this is that each model may be good for different reason. Therefore, ensembling will provide “the best of both worlds” and result in a model superior to each of the constituent parts. The ensembling can take a naive form of merely equally weighing every predictor or it can be data-based, in which the validation data dictates the weights which provide the best combination. Most state-of-the-art machine learning models that score well in competitions end up relying on some form of ensembling.

**Maximum Likelihood Estimation**

*Maximum likelihood estimators* are special cases of *M-estimators*. Consequently, we will first review some general theorems concerning *M-estimators*. One of the most important concepts in estimation is the idea of consistency:

**Definition 2.1** (Consistency). The estimator \( \hat{\theta}_n \) used to estimate the parameter \( \theta \) is called *asymptotically consistent* if the sequence \( \hat{\theta}_n \) converges in probability to \( \theta \). That is, if \( d \) is some distance metric, \( \mathbb{P}[d(\hat{\theta}_n, \theta) \geq \varepsilon] \to 0 \) as \( n \to \infty \) for every \( \varepsilon > 0 \). We write \( \hat{\theta}_n \overset{P}{\to} \theta \).

Following van der Vaart [41], suppose that the M-estimator \( \hat{\theta}_n \) maximizes the random criterion function \( M_n(\theta) \). Under suitable normalization, there exists a function \( M(\theta) \) such that \( M_n(\theta) \overset{P}{\to} M(\theta) \) for every \( \theta \). Then, we would expect that if \( \theta_n \) is a maximizer of \( M_n \), it will converge to the maximizing value \( \theta_0 \) of \( M \). However this is not the case, and we need more than the convergence in probability of the random criterion functions.

**Theorem 2.7.** Let \( M_n \) be random functions and let \( M \) be a fixed function of \( \theta \) such that such that for every \( \varepsilon > 0 \) (probability statements should be understood in terms of
Then, any sequence of estimators \( \hat{\theta}_n \) with \( M_n(\hat{\theta}_n) \geq M_n(\theta_0) - o_P(1) \) converges in probability to \( \theta_0 \).

**Proof.** We have \( M_n(\hat{\theta}_n) \geq M_n(\theta_0) - o_P(1) \). Since \( M_n \) converges uniformly to \( M \), then \( M_n(\theta_0) \xrightarrow{P} M(\theta_0) \), so the right side equals \( M(\theta_0) - o_P(1) \). It follows \( M_n(\hat{\theta}_n) \geq M(\theta_0) - o_P(1) \), since

\[
M(\theta_0) - M(\hat{\theta}_n) \leq M_n(\hat{\theta}_n) - M(\theta_0) + o_P(1)
\]

\[
\leq \sup_{\theta} |M_n - M|(\theta) + o_P(1) \xrightarrow{P} 0
\]

using the first part of the theorem assumption. Using the second part of the assumption, there exists for every \( \varepsilon > 0 \) a number \( \eta > 0 \) such that \( M(\theta) < M(\theta_0) - \eta \) for every \( \theta \) with \( d(\theta, \theta_0) \geq \varepsilon \). Therefore, the event \( \{d(\hat{\theta}_n, \theta_0) \geq \varepsilon \} \) is contained in the event \( \{M(\hat{\theta}_n) < M(\theta_0) - \eta \} \). The probability of the latter converges to 0 as previously established.

Having seen that the \( M \)-estimator is asymptotically consistent under appropriate conditions, one might reasonably ask about the rate at which the discrepancy \( \hat{\theta}_n - \theta \) converges to zero. While the answer may be situation-specific, for estimators based on \( n \) replications of an experiment, this rate is often \( n^{-1/2} \). We can now multiply with the reciprocal to re-establish balance and, it turns out, the sequence \( \sqrt{n}(\hat{\theta}_n - \theta) \) oftentimes converges in distribution to a normal distribution. This is particularly useful, for example, when creating approximate confidence intervals.

For a detailed treatment of asymptotic normality in \( M \)-estimators, we refer the reader to confer Theorems 5.21 and 5.23 in van der Vaart [41] and will provide here a
non-rigorous treatment. First, we point out that finding \( \theta \), the maximizer of \( M_n(\theta) \) over \( \Theta \) typically proceeds by setting derivatives equal to 0. The resulting estimating equations are called Z-estimators:

\[
\Psi_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \psi_{\theta}(X_i) = 0,
\]

where \( \psi_{\theta} \) are known vector-valued maps, a shorthand for the corresponding system of equations resulting from the maximization process.

We will now use this characterization of \( M \)-estimators in terms of the estimating equation and consider \( X_1, \ldots, X_n \) be a sample from some distribution \( P \), and let the random “true” criterion function take the form:

\[
\Psi_n(\theta) \equiv \frac{1}{n} \sum_{i=1}^{n} \psi_{\theta}(X_i) = \mathbb{P}_n \psi_{\theta}
\]

\[
\Psi(\theta) = \int \psi_{\theta} dP
\]

We now assume that the estimator \( \hat{\theta}_n \) is a zero of \( \Psi_n \) and converges in probability to \( \theta_0 \), a zero of \( \Psi \). It, therefore makes sense to perform Taylor expansion of \( \Psi_n(\hat{\theta}_n) \) around \( \theta_0 \). If \( \theta \) is one-dimensional, then

\[
0 = \Psi_n(\hat{\theta}_n) = \Psi_n(\theta_0) + (\hat{\theta}_n - \theta_0) \Psi_n'(\theta_0) + \frac{1}{2}(\hat{\theta}_n - \theta_0)^2 \Psi_n''(\tilde{\theta}_n),
\]

where \( \tilde{\theta}_n \) is a point between \( \hat{\theta}_n \) and \( \theta_0 \). Then, we can get by re-writing

\[
\sqrt{n}(\hat{\theta}_n - \theta_0) = \frac{-\sqrt{n}\Psi_n'(\theta_0)}{\Psi_n'(\theta_0) + \frac{1}{2}(\hat{\theta}_n - \theta_0) \Psi_n''(\tilde{\theta}_n)}
\]

If \( \psi_{\theta_0} \) is square-integrable under the measure \( P \), the numerator \( -\sqrt{n}\Psi_n'(\theta_0) \) is asymptotically normal by the central limit theorem, with mean \( \int \psi_{\theta_0} dP = \Psi(\theta_0) = 0 \) and variance \( \int \psi_{\theta_0}^2 dP \). Focusing on the denominator, it is easy to see that \( \Psi_n'(\theta_0) \xrightarrow{P} \int \psi_{\theta_0} dP \) by the law of large numbers, provided the expectation exists. Under the reasonable condition that \( \tilde{\Psi}_n(\tilde{\theta}_n) \) is \( O_P(1) \) and considering that \( \hat{\theta}_n - \theta_0 = o_P(1) \), we use Slutsky’s theorem to conclude that

\[
\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow{d} \mathcal{N} \left( 0, \frac{\int \psi_{\theta_0}^2 dP}{\left( \int \psi_{\theta_0} dP \right)^2} \right)
\]
It turns out that a sufficient regularity condition on $\psi$ is that the latter is Lipschitz with constant $\dot{\psi}(x)$, which provides a good compromise between simplicity and general applicability.

We are now equipped to turn to maximum likelihood estimators, which are examples of $M$-estimators. First, one must impose a condition of identifiability on the model.

**Definition 2.2 (Identifiable).** For every parameter $\theta \neq \theta_0$, we have $P_\theta \neq P_{\theta_0}$. That is, the model for the observations is not the same under the parameters $\theta$ and $\theta_0$. 

If $X_1, \ldots, X_n$ are a random sample from a density $p_\theta$, then the maximum likelihood estimator $\hat{\theta}_n$ maximizes the function

$$M_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{p_\theta}{p_{\theta_0}}(X_i).$$

Note that the extra $\sum \log p_{\theta_0}(X_i)$ is added for mathematical convenience. It is easy to see that the asymptotic function for $M_n$ is

$$M(\theta) = E_{\theta_0} \log \frac{p_\theta}{p_{\theta_0}}(X)$$

The question is whether the value $\theta_0$ is always a point of maximum of $M(\theta)$. The following lemma provides the answer.

**Lemma 2.8.** Let $\{p_\theta : \theta \in \Theta\}$ be a collection of subprobability densities such that the model is identifiable and and $P_{\theta_0}$ is a probability measure. Then $M(\theta) = E_{\theta_0} \log \frac{p_\theta}{p_{\theta_0}}(X)$ attains its maximum uniquely at $\theta_0$.

**Proof.** First, we observe that $M(\theta_0) = 0$. Write $\mu$ for the dominating measure and use the inequality $\log x \leq 2(\sqrt{x} - 1)$ for every $x \geq 0$ and we have

$$E_{\theta_0} \log \frac{p_\theta}{p_{\theta_0}}(X) \leq 2 E_{\theta_0} \left( \sqrt{\frac{p_\theta}{p_{\theta_0}}} - 1 \right)$$

$$= 2 \int \sqrt{p_\theta p_{\theta_0}} d\mu - 2$$

$$\leq - \int (\sqrt{p_\theta} - \sqrt{p_{\theta_0}})^2 d\mu.$$
Incidentally, the last inequality is an equality if $p_\theta$ is a probability measure. We note that the quantity in the last display is nonpositive and is zero if and only if $p_\theta = p_{\theta_0}$. By assumption, it follows that $\theta = \theta_0$.

This section explored properties of the maximum likelihood estimator and showed conditions for asymptotic consistency and normality and showed that under regularity conditions, the estimator is, in some sense, asymptotically efficient. This discussion will frame the use of the maximum likelihood estimator in the survival analysis problem to follow.

**Gaussian Mixture Models**

The Gaussian distribution is arguably the most prevalent distribution in statistics in part due to the many analytical properties it possesses. However, when modeling with real data sets, it suffers from significant limitations. In particular, when the distribution of the real data appears multimodal, a mixture distribution may be more appropriate. In this section, we focus on a particular kind of mixture distribution, namely, a *Gaussian mixture model*. We begin with a few definitions and then address maximum likelihood estimation in the context of Gaussian mixture models by discussing the Expectation Maximization (EM) algorithm.

**Definition 2.3** (Multivariate Normal). Let $Z \in \mathbb{R}^d$ and $Z = (Z_1, \ldots, Z_d)^T$, where the $Z_j$ are independent standard normal variables. Then, the random vector $U \in \mathbb{R}^d$ has a multivariate normal distribution if and only if there exist constants $\mu \in \mathbb{R}^d$ and $A \in \mathcal{M}^{d \times d}(\mathbb{R})$ such that $U$ can be written as

$$U = \mu + AZ.$$ 

In this case, we write $U \sim \mathcal{N}_d(\mu, \Sigma)$, where $\Sigma = AA^T$ is the variance-covariance matrix.
Using straightforward calculations, we then have for the density of \( U \)

**Theorem 2.9.** If \( \Sigma \) is a positive definite matrix, then, if \( U \sim \mathcal{N}_d(\mu, \Sigma) \), \( U \) has density given by

\[
p_U(x) = \frac{1}{(2\pi)^{d/2} |\text{det}(\Sigma)|^{d/2}} \times \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\}
\]

We now define a mixture of Gaussians.

**Definition 2.4** (Mixture of Gaussians). Let \( \pi_1, \ldots, \pi_K \) for a positive integer \( K > 1 \) be parameters called *mixing coefficients* such that \( \sum_{k=1}^{K} \pi_k = 1 \) and \( 0 < \pi_k < 1 \) for all \( k \) in \( 1, \ldots, K \). Then, the probability density function of a *mixture of \( K \) Gaussians* is given by

\[
p(x) = \sum_{k=1}^{K} \pi_k \phi_{\mu_k, \Sigma_k}(x),
\]

where each \( \phi_{\mu_k, \Sigma_k} \) is a multivariate normal distribution with mean \( \mu_k \) and variance-covariance matrix \( \Sigma_k \) as given above.

Now, consider a sample \( S_1, \ldots, S_n \) from the Gaussian mixture population above and consider the log likelihood for this problem. We have

\[
\ln p(S|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \ln \left\{ \sum_{k=1}^{K} \pi_k \phi_{\mu_k, \Sigma_k}(S_i) \right\},
\]

where \( \pi = \{\pi_1, \ldots, \pi_K\} \), \( \mu = \{\mu_1, \ldots, \mu_K\} \), and \( \Sigma = \{\Sigma_1, \ldots, \Sigma_K\} \).

Clearly, if the goal is to estimate the parameters using maximum likelihood, the situation is much more complicated than in the single Gaussian distribution case. It is not hard to see that on the boundary of the parameter set, the likelihood can tend to \( \infty \) and, therefore a maximum likelihood estimator does not exist. However, it turns out that the EM algorithm can provide a good proxy via a local maximum.

**The EM Algorithm**

The description of the EM algorithm here follows the presentation in [43]. Assume there are ideal observations \( X \sim P_\theta \) with density \( p(x, \theta) \) for some parameter \( \theta \) in the
parameter space $\Theta \subset \mathbb{R}^d$, which lead to a log likelihood function $\ell_{p,x}(\theta)$ which is easily maximized. For example, the log likelihood could be concave in $\theta$ and have a unique closed-form maximum likelihood estimator. Unfortunately, the actual observed data is some $S \equiv S(X) \sim Q_\theta$ with density $q(s, \theta)$, where $\ell_{q,s}(\theta) = \log q(s, \theta)$ is difficult to maximize. One useful way to look at this setup is to see $S$ as representing a part of $X$, where the other part of $X$ is “missing”. The process of “reconstructing” $X$ is the process of estimating $\theta$ by the maximum likelihood.

It is not hard to see that the case of mixture of Gaussians falls under the setup described above. For example, as before, let $S_1, \ldots, S_n$ be a sample from a population $P$ whose density is modeled as a mixture of $K$ Gaussians. In this case, $\theta = \{\pi, \mu, \Sigma\}$. If we let

$$X_i = (\Delta_i, S_i), \quad 1 \leq i \leq n,$$

where $\Delta_i$ are independent identically distributed with $P_\theta\{\Delta_i = 1\} = \pi_i = 1 - P_\theta\{\Delta_i = 0\}$. It is not hard to see that conditioned on $\Delta$, the $S_i$’s will follow a normal distribution. So one can think of $S$ as $S(X)$, where $X$ is given above. We are now ready to state the algorithm in general terms.

**General EM Algorithm.** Given a joint distribution $p(S, \Delta|\theta)$ over observed variables $S$ and latent variables $\Delta$, governed by parameters $\theta$, the goal is to maximize the likelihood function $q(S|\theta)$.

1. Initialize the value of the parameters $\theta_{old}$.

2. **E step** Evaluate $p(\Delta|S, \theta_{old})$

3. **M step** Evaluate $\theta_{new}$ given by

$$\theta_{new} = \arg \max_\theta Q(\theta, \theta_{old})$$
where the expectation $Q$ is given by

$$Q(\theta, \theta_{old}) = \sum_{\Delta} p(\Delta | S, \theta_{old}) \ln p(S, \Delta | \theta)$$

4. If the log likelihood or the parameter values converged, return $\theta_{new}$. Otherwise, let

$$\theta_{old} \leftarrow \theta_{new}$$

and return to step 2.

To see the algorithm applied to a mixture a Gaussians, we refer the reader to [11], Chapter 9. It turns out that for distributions in the exponential family, if the maximum in the M step exists, then it is necessarily unique [43]. Furthermore, if the sequence of iterates $\theta_j$ is bounded and the expectation equation has a unique solution, then the sequence converges and the point to which it converges is necessarily a local maximum of the distribution sought.

**Survival Analysis**

One of the central problems in this work will be to analyze survival data for patients at risk of sudden cardiac death from arrhythmia and build a survival model using neural networks. In this section, we set up the mathematics behind the survival analysis problem. First, we define basic terminology, then we discuss the problem of missing data (censoring). Later, we review examples of non-parametric, semiparametric and parametric methods of modeling survival analysis data.

Survival analysis does not have a precise definition, but a common theme is the analysis of positive-value random variables using various statistical techniques [44]. The variable of interest is typically time to some event. For example, the event could be the time to failure of some physical mechanical or electrical component (e.g., time until the filament of a light bulb burns out, or time until a steel beam ruptures due
to load. In biological applications, the time to failure usually refers to death of a particular unit (e.g., cell, animal, patient, etc.). The random variable of interest can also be the time it takes to acquire a certain skill or can be something entirely different than time, like the payout of a financial instrument (e.g., option contract) as it gets close to maturity.

We will primarily focus the analysis on medical applications. In this case, where the event of interest is death due to a particular condition (e.g., prostate cancer), the problem is complicated by the existence of competing risks. For example, if the patient population under analysis is geriatric, then heart failure could be considered a competing risk to the analysis of prostate cancer mortality. In this case, cause-specific survival probabilities may be of less medical interest when the survival probability depends on several other factors. This scenario will be addressed at the end of this section.

**Problem Setup**

Let $T$ be a random variable such that $T \geq 0$ with distribution function $F(t)$. We assume that the probability density function exists and denote it by $f(t)$.

**Definition 2.5** (Survival Function). Let $T$ be a random variable as above. Then, the survival function is given by

$$S(t) = 1 - F(t) = P\{T > t\}.$$ 

Another important concept in survival analysis which is oftentimes modeled directly is the hazard function (or hazard rate).

**Definition 2.6** (Hazard Rate). Under the conditions on $T$ given above, the hazard rate or hazard function is given by

$$\lambda(t) = \frac{f(t)}{S(t)}.$$
The hazard rate $\lambda(\cdot)$ has a straightforward interpretation as seen by the following heuristic expression:

$$\lambda(t)dt = \Pr\{t < T < t + dt|T > t\}$$

That is, $\lambda(t)$ is the probability of expiring in the infinitesimal interval $(t, t + dt)$ conditioned on having survived past time $t$. An important relationship between the survival function and the hazard rate emerges by considering that

$$S(t) = e^{-\int_0^t \lambda(u)du}$$

We also note that

**Proposition.** With the distribution function, survival function and hazard rate defined as above, we have as $t \to \infty$

$$F(t) \to 1 \text{ iff } S(t) \to 0 \text{ iff } \int_0^t \lambda(u)du \to \infty.$$
it. The “random” part of the censoring comes from the following model we adopt for the censoring times.

Let $C_1, \ldots, C_n$ be independent and identically distributed times, each with distribution function $G$. Associated with each event time $T_i$, there is a censoring time $C_i$. However, the actual observed variables are the pairs $(X_1, \Delta_1), \ldots (X_n, \Delta_n)$, where

$$X_i = \min\{T_i, C_i\}$$

$$\Delta_i = 1\{T_i < C_i\} = \begin{cases} 1 & \text{if } T_i \text{ is not censored,} \\ 0 & \text{if } T_i \text{ is censored} \end{cases}$$

A typical assumption in random censoring (which we will relax later) is that the event times $T_i$ and censoring times $C_i$ are statistically independent. Then, the distribution of a typical observation $(X_i, \Delta_i)$ satisfies

$$P_{F,G}(X \leq x, \Delta = 0) = \int_{[0,x]} (1 - F) dG,$$

$$P_{F,G}(X \leq x, \Delta = 1) = \int_{[0,x]} (1 - G_-) dF,$$

where the minus sign in the subscript denotes the left continuous version of the function. Then, if $F$ and $G$ have densities $f$ and $g$ (relative to some dominating measures), the joint r.v. $(X, \Delta)$ has density

$$(x, \delta) \mapsto [(1 - F)(x) g(x)]^\delta [(1 - G_-)(x) f(x)]^{1-\delta} \quad (2.10)$$

**Estimation**

We now focus on estimating the survival function $S(t)$ or its related quantities. It can be seen from $(2.10)$ that if $f$ and $g$ are interpreted as Lebesgue densities, this expression cannot be used as part of a likelihood. Consider, for example, an $f$ with a high spike at an observation $X_i$, which would make the supremum infinity. In what follows, we will provide an overview of some common nonparametric, semiparametric,
and parametric approaches for estimating the survival function. The discussion that follows on nonparametric and semiparametric ways to estimate the survival function is based on the work in [41].

**Nonparametric Estimation**

We begin the discussion on estimation with the following definition.

**Definition 2.7 (Cumulative Hazard Function).** The cumulative hazard function corresponding to a cumulative distribution function $F$ on $[0, \infty]$ is defined as

$$\Lambda_F(t) = \int_{[0,t]} \frac{dF}{1 - F}.$$

We note that if $F$ has a density $f$, then $\Lambda_F$ has a density $\lambda_F = f/(1 - F)$, which is precisely the aforementioned hazard rate. Given this, we can interpret $d\Lambda_F(t)$ as the probability of “instant death at time $t$ given survival until $t$” as before. It turns out that the hazard function is a very important concept in survival analysis, even more so when considering that there exists a one-to-one correspondence between it and the distribution function. In Lemma 25.74, van der Vaart shows [41] that the cumulative distribution function can be explicitly recovered from the cumulative hazard function as the product integral of the $-\Lambda$. Specifically, if we let $\Lambda\{s\}$ be the jump of $\Lambda$ at $s$ and $\Lambda^c(s)$ be the continuous part, we have

$$1 - F_\Lambda(t) = \prod_{0<s\leq t} \left(1 - \Lambda\{s\}\right) e^{-\Lambda^c(t)}.$$  \hspace{1cm} (2.11)

Importantly, Lemma 20.14 in van der Vaart [41] shows that, under some restrictions, the maps $F \leftrightarrow \Lambda_F$ are Hadamard differentiable, which means that, from an asymptotic-statistical point of view, estimating a cumulative hazard function and estimating a distribution function are the same problem.

In light of the previous discussion, we now turn to the problem introduced before of estimating a distribution function based on right-censored data. Instead of focusing on
we turn instead to the cumulative hazard function $\Lambda$, where we drop the subscript $F$ for legibility. It is easy to see that if we let

$$1 - H = (1 - F)(1 - G)$$

$$dH_1 = (1 - G_-)dF$$

for every choice of censoring distribution $G$, then the cumulative hazard of interest can be written as

$$\Lambda(t) = \int_{[0,t]} \frac{1}{1 - F_-} dF = \int_{[0,t]} \frac{1}{1 - H_-} dH_1$$

Under the same independent censoring assumption we made before — that is, that the event times $T_i$ and censoring times $C_i$ are statistically independent, then $H$ is precisely the distribution function of $X_i$ and $H_1$ is a “subdistribution function”:

$$1 - H(x) = \mathbb{P}\{X_i > x\}$$

$$H_1(x) = \mathbb{P}\{X_i \leq x, \Delta_i = 1\}$$

We can now obtain an estimator for $\Lambda$ by estimating each one of these two functions by the empirical distribution of the data. That is, let

$$\hat{H}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{X_i \leq x\}$$

$$\hat{H}_{1n}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{X_i \leq x, \Delta_i = 1\}.$$ 

These estimators can now be used directly in the formula for $\Lambda$ and we get the famous *Nelson-Aalen estimator* of the cumulative hazard rate:

$$\hat{\Lambda}_n(t) = \int_{[0,t]} \frac{1}{1 - \hat{H}_{1n}} d\hat{H}_n.$$ 

It is important to note that the maps used to construct the Nelson-Aalen estimator from $\hat{H}_n$ and $\hat{H}_{1n}$ are Hadamard differentiable, which means that the estimator inherits their properties. Specifically, since they are empirical distributions, their sequence is asymptotically normal and so will be that for the Nelson-Aalen estimator.
Using the Nelson-Aalen estimator, we can directly obtain an estimator for the survival function $\hat{S}_n = 1 - \hat{F}_n$ in light of the product integral (2.11) to obtain the Kaplan-Meier estimator

$$1 - \hat{F}_n(t) = \prod_{i: X_i \leq t} \frac{\#(j : X_j \geq X_i) - \Delta_i}{\#(j : X_j \geq X_i)} = \prod_{i: X_{(i)} \leq t} \left( \frac{n - i}{n - i + 1} \right)^{\Delta_{(i)}}. \quad (2.12)$$

Since the product integral is Hadamard differentiable, it follows that the Kaplan-Meier estimator sequence is asymptotically normal.

**Semiparametric Estimation**

In semiparametric models, the universe of possible parameters is not as “vast” as in nonparametric ones, where parameters range over “infinite-dimensional” sets, but also not quite as “nicely-behaved” as in the parametric case, where the parameters of interest are typically a vector in Euclidean space. For example, the semiparametric model of interest might have a natural parametrization given by $(\theta, \eta) \mapsto \mathbb{P}_{\theta, \eta}$, where $\theta$ is a Euclidean vector and $\eta$ runs through a nonparametric class of distributions, or some infinite-dimensional set. Typically, in this setting we aim at estimating $\theta$, whereas $\eta$ is considered a *nuisance* parameters.

Maximum likelihood estimators arising in semiparametric models are oftentimes not dominated or are defined using densities with maxima going to infinity. This means that the functions chosen to represent the “likelihoods” need to be carefully selected. Assume the likelihood for the parameter $P$ given one observation $x$ be given by $\mathcal{L}(P)(x)$. Now consider a partitioned parameter $(\theta, \eta)$. It is sometimes useful to consider the *profile likelihood* defined for a likelihood $\mathcal{L}_n(\theta, \eta)(X_1, \ldots X_n)$ as the function

$$\theta \mapsto \sup_\eta \mathcal{L}_n(\theta, \eta)(X_1, \ldots X_n)$$

Here, the supremum is taken over all possible values of the parameter $\eta$ in its respective
infinitely-dimensional space. Note that this is equivalent to maximizing in two steps over \( \eta \), then \( \theta \). In practice, these profile likelihoods are used as proper likelihood functions of parametric models. Next, we see how these ideas apply in the context of a popular semiparametric model from survival analysis.

**Cox proportional hazards.** A tremendously popular semiparametric model in survival analysis is the *Cox proportional hazards*. We will discuss this in the absence of censoring, with the understanding that the extension follows easily. Consider the observation \( X = (T, Z) \), where \( T \) is the “survival time” as before and \( Z \) is a vector of covariates. We assume that the distribution of \( Z \) is totally unknown and the conditional hazard function of \( T \) given \( Z \) is given by

\[
\lambda_{T|Z}(t) = e^{\theta^T Z} \lambda(t),
\]

where \( \lambda \) is a completely unspecified hazard function, typically called *baseline hazard function*. The density of the observation \( X = (T, Z) \) is given by

\[
e^{\theta^T z} \lambda(t) e^{-e^{\theta^T z} \Lambda(t)},
\]

where \( \Lambda \) is the cumulative hazard function with \( \Lambda(0) = 0 \). The likelihood in this model is given by, with \( \Lambda\{t\} \) the jump of \( \Lambda \) at \( t \),

\[
(\theta, \Lambda) \mapsto \prod_{i=1}^{n} e^{\theta^T z_i} \Lambda\{t_i\} e^{-e^{\theta^T z_i} \Lambda(t_i)}.
\]

Note that this expression is very close to the usual product of densities, with the only change that the hazard function \( \lambda(t) \) is replaced by the jumps \( \Lambda\{t\} \) of the cumulative hazard function, effectively equivalent to viewing \( \lambda(t) \) as piece-wise constant in between event times \( t_i \). Maximum likelihood estimators \( (\hat{\theta}, \hat{\Lambda}) \) can now be derived in closed-form by “profiling out” the nuisance parameter \( \Lambda \). First, we write \( \lambda_i = \Lambda\{t_i\} \) for all \( i = 1, \ldots, n \). We find these \( \lambda_i \)'s by easily observing that the function

\[
(\lambda_1, \ldots, \lambda_n) \mapsto \prod_{i=1}^{n} e^{\theta^T z_i} \lambda_i e^{-e^{\theta^T z_i} \sum_{j:t_j \leq t_i} \lambda_j}
\]

47
is maximized for

\[
\frac{1}{\lambda_k} = \sum_{i: t_i \geq t_k} e^{\theta^T z_i}
\]

In view of this expression for the jumps in the cumulative hazard function, we take the supremum over \( \Lambda \) to obtain the profile likelihood

\[
\theta = \mapsto \prod_{i=1}^n \frac{e^{\theta^T z_i}}{\sum_{j: t_j \geq t_i} e^{\theta^T z_j}},
\]

which is known as the Cox partial likelihood (note that constant terms were dropped). This criterion was originally motivated using conditional probabilities that the \( i \)th subject dies at time \( t_i \) given that one of the subjects at risk dies at that time. We can now also get the maximum likelihood estimator for \( \Lambda \) as the step function with jumps

\[
\hat{\Lambda}\{t_k\} = \frac{1}{\sum_{i: t_i \geq t_k} e^{\theta^T z_j}}
\]

The estimator in the last display is sometimes called the Breslow estimator [45]. Finally, we note that, in order to incorporate censoring, only a small correction is needed to the likelihood

\[
(\theta, \Lambda) \mapsto \prod_{i=1}^n \left\{ e^{\theta^T z_i \Lambda\{t_i\}} \right\}^{\delta_i} e^{-e^{\theta^T z_i \Lambda(t_i)}}
\]

and the rest of the analysis proceeds in a similar fashion.

**Parametric Estimation**

Perhaps the most straightforward way to estimate survival characteristics is to use parametric models. The estimation process begins by making some distributional assumptions on the data. For example, assume that the times to failure \( T_i \) are independent and identically distributed according to the Weibull distribution with shape \( k \) and scale \( s \), such that the survival function takes the form

\[
S(t; k, s) = e^{-(t/s)^k}
\]
The, one can write the likelihood

\[(k, s) \mapsto \prod_{i=1}^{n} S(x_i; k, s)^{\delta_i} f(x_i; k, s)^{1-\delta_i}\]

in view of (2.10) and, once again, assuming independent censoring, this likelihood can be maximized independently of the portion related to the censoring distribution to find \(k\) and \(s\).

**Competing Risk Analysis**

Up to this point, the main assumption behind the survival models discussed has been that the censoring process is statistically independent from the survival times. While this may be a good assumption when the failure time is that of a mechanical or electrical component, this modelling assumption is not suited for many medical applications. As Fine and Gray point out [46, 47], models relying on the independent censoring assumptions are inadequate when the time to failure of interest is the time to a specific disease (e.g., “time to death from lung cancer”). In such cases, competing risks are present, for example death from any other causes, which not only may not be statistically independent to the one of interest, but make estimated cause-specific survival probabilities less useful in practice. In such settings, the measure of interest to be estimated is the *cumulative incidence function*.

The general setting remains similar, but we assume that there are now multiple causes of death denoted by \(K = 1, \ldots, k\) that are mutually exclusive. In this setting, the observed variable will be similar \(X = \min\{T_1, \ldots, T_k, C\}\), but also include \(K\) the cause of death. We now define the quantity we wish to estimate [48]

**Definition 2.8 (Cumulative Incidence Function).** The *cumulative incidence function* in a competing risk survival analysis problem is given by

\[C_k(t) = \mathbb{P}\{T < t, K = k\}\]
Alternatively, one can write the cumulative incidence function as

\[ C_k(t) = \int_{[0,t]} S(u)\lambda_k(u)du, \]

where \( \lambda_k \) is the hazard rate associated with a particular cause of death. There are several approaches used to estimate the cumulative incidence rate, but we will focus on two common ones. One of the more straightforward methods is the *cause-specific* approach, where the hazards are defined by

\[ \lambda_{CS}^k(t) = \lim_{h\to 0} \frac{P\{t < T \leq t + h, K = k | T \geq t\}}{h} \]

Using a Cox proportional hazards, one can estimate the hazards \( \lambda_{CS}^k(t) \) and use the Kaplan-Meier estimator given by Eq. (2.12) for the total survival function \( S(t) \) to obtain an estimate for the cumulative incidence function. Alternatively, one can use a maximum likelihood approach by parametrizing the \( \lambda_{CS}^k(t) \) with a suitable distribution. Alternatively, the incidence function can be parametrized directly [49].

Another popular approach in competing risk analysis is to use the *subdistribution cumulative incidence*, which arises from the hazard rate [48]

\[ \lambda_{SD}^k(t) = \lim_{h\to 0} \frac{P\{t < T \leq t + h, K = k | T > t \text{ or } (T < t \text{ and } K \neq k)\}}{h} \]

A Cox proportional hazards with risk set modifications and weighing is the standard regression approach for estimating the cumulative incidence function as previously described [47].
Chapter 3

Anatomically-Informed Deep Learning on Contrast-Enhanced Cardiac MRI for Scar Segmentation and Clinical Feature Extraction

Introduction

Many cardiac diseases are associated with structural remodeling of the myocardium. In both ischemic and non-ischemic cardiomyopathies, the presence of myocardial fibrosis and scar significantly elevates the risk for lethal heart rhythm disorders and sudden cardiac death (SCD) [50, 51, 52]. Therefore, assessment of myocardial scar and fibrosis is important for diagnostic and prognostic purposes, in forecasting the trajectory of heart disease [53], evaluating arrhythmia propensity in the heart [54, 55], and stratifying patients for SCD risk [56, 57].

Cardiac magnetic resonance (CMR) imaging with late gadolinium enhancement (LGE) has unparalleled capability in the detection and quantification of scar and fibrosis due to increased brightness in regions with a higher proportion of extracellular space [58]. The utility of scar/fibrosis assessment in clinical decision-making has been demonstrated by a large body of clinical research in patients with different...
cardiomyopathies [55, 59, 60, 61, 62, 52, 51, 57], and by a number of mechanistic studies of arrhythmogenesis in heart disease [63, 64, 65, 66]. However, LGE-CMR image analysis is a laborious task prone to substantial inter-observer variability. It requires expert contouring of the epicardial and endocardial borders, the intermediate-intensity peri-infarct zone (gray zone, GZ), and the high-intensity dense scar region. There is an unmet need for an automated method to segment myocardium and scar in LGE-CMR images. Ideally, these segmentations should follow anatomical guidelines to ensure seamless extraction of important clinical features used in patient prognosis.

Deep learning (DL)-based image segmentation offers the promise of full automation and consistency of output. However, most of the available algorithms still require intensive manual interventions, e.g., specifying anatomical landmarks [67] or labeling boundary slices of the stack at the apex and base of the heart [68]. The few DL algorithms developed for LGE-CMR myocardial segmentation [69, 70, 71, 72] and the even fewer for LGE-CMR scar/fibrosis segmentation [73, 74, 75, 76] all suffer from significant limitations. Specifically, these approaches are not robust to varying image acquisition quality (i.e., different scanners and protocols at different centers) or to the varying fibrosis patterns resulting from different heart pathologies, leading to bespoke algorithms which fail to generalize across populations [73, 75, 76].

Here we develop an anatomically-informed DL approach to LGE-CMR image segmentation and clinical feature extraction. We term our technology Anatomical Convolutional Segmentation Network, ACSNet. This fully automated technology applies three stages of deep neural networks to segment the left ventricle (LV), contour the LV myocardium, blood pool, and scar/fibrosis regions, and apply geometric constraints to the segmentations to ensure anatomical accuracy. ACSNet is robust to different scar/fibrosis distributions, to inputs from various imaging centers acquired on scanners from different manufacturers, and to multiple CMR modalities. It outperforms inter-expert segmentation results and demonstrates consistently accurate performance
across often ambiguous regions of the LV (e.g., apex and base). Segmentations satisfy anatomical guidelines, allowing for expert-level computation of clinical features, such as scar burden and LV volume.

Methods

Imaging Data and Processing

The primary data source for ACSNet was 2-D LGE-CMR scans acquired during two clinical trials: Left Ventricular Structural Predictors of Sudden Cardiac Death (ClinicalTrials.gov ID NCT01076660) and Prospective Observational Study of Implantable Cardioverter Defibrillators (PROSE-ICD, NCT00733590). All LGE-CMR images used in this study were acquired using 1.5-T MRI devices (Signa, GE Medical Systems, Waukesha, Wisconsin; Avanto, Siemens, Erlangen, Germany). The contrast agent used was $0.15 - 0.20$ mmol/kg gadodiamide (Omniscan, GE Healthcare) and the scan was captured 10–30 minutes after injection. The most commonly used sequence was inversion recovery fast gradient echo pulse, with an inversion recovery time typically starting at 250ms and adjusted iteratively to achieve maximum nulling of normal myocardium. Typical spatial resolutions ranged $1.5 - 2.4 \times 1.5 - 2.4 \times 6 - 8$ mm, with 2–4mm gaps. After excluding scans with very poor quality, 1,124 2-D LGE-CMR slices were selected from 155 patients with ischemic cardiomyopathy (ICM). Trained experts provided manual segmentations of myocardium and scar/fibrosis as described in previous work [77].

LGE data was supplemented with 1,360 2-D short-axis end diastole (ED) cine CMR slices (245 scans) from two publicly available sources: MICCAI Automated Cardiac Diagnosis Challenge[78] and Cardiac MR Left Ventricular Segmentation Challenge [79]. Ground truth myocardium segmentations were provided with the scans. The cine CMR data set was converted into “LGE-like” images using a custom
style transfer method (fig. 3-1). First, a pseudo-enhancement mask was generated by intersecting the myocardium mask with a randomized collection of basic shapes (e.g., ellipses) with random locations. The resulting patches were then blurred using Gaussian filters, leading to smoother edges. The resulting mask was overlaid onto the original (dark) myocardium, elevating the signal intensity in the corresponding area. Then, speckle noise was added and, finally, for each cine scan, a histogram match was performed between the newly generated image and randomly sampled scans from the LGE training data set.

Figure 3-1. Conversion Process to “LGE-like” Cine Images. (1) The original cine image is cropped/padded to a square and contrast-limited adaptive histogram equalization (CLAHE) is applied. (2) Cine images are further transformed by first generating a pseudo-enhancement (“LGE-like” enhanced myocardium) mask. (3) We apply random erosion of the pseudo-scar mask and Gaussian filter to realistically blur the edges. (4) Next, speckle noise is added to the image to resemble LGE noise. (5) Finally, we match the histogram to that of LGE-CMR scan sampled randomly.

All LGE and resulting “LGE-like” 2-D slices were pre-processed and stored in a common file format to accommodate multiple medical image file types (e.g., DICOM, NIfTI, etc.), retaining 3-D ventricular geometry information. Specifically, slices
were ordered apex to base, retaining slice location, image intensities, resolution, and patient orientation information. Slices with no ground truth myocardial segmentation were excluded from training. The images were approximately standardized in terms of orientation by applying rotations in increments of 90° (90° was chosen to avoid interpolation). If scans originally stored in DICOM had the “WindowCenter”, “WindowLength”, “RescaleSlope”, and “RescaleIntercept” tags populated, the corresponding linear transformation was applied to the raw signal intensities to enhance contrast and brightness.

Finally, to increase the contrast between myocardium and blood pool, contrast-limited adaptive histogram equalization (CLAHE) [80] was applied. All images were cropped or padded to a square of size 192 × 192 pixels (no aspect ratio distortion), without centering. Finally, resulting images were re-scaled in the range [0, 255].

Model

The overall architecture of our DL approach is presented in fig. 3-2. ACSNet used a multi-network sequential approach to segment viable myocardium, enhanced myocardium (scar/fibrosis), and blood pool. The first network identified the LV region of interest (ROI), which was then used to zero out signal outside a tight square around the segmented ROI (fig. 3-2A). The second network differentiated between myocardium and enhancement (fig. 3-2B). The third was a post-processing stage, which adjusted the predictions to satisfy anatomical constraints (fig. 3-2C,D).

Stage 1: Region of Interest Segmentation Network

The first network in ACSNet (fig. 3-2A) was trained to predict a mask of the LV region of interest (ROI), which included myocardium and blood pool. The goal of this network was to simplify the detailed segmentation problem in the next stage by reducing the very high ratio of background-to-myocardium pixels and limiting
Figure 3-2. ACSNet architecture consisting of three inter-connected DL sub-networks. (A) The first residual U-Net (ResU-Net) is used to identify and crop around the left ventricle (LV). (B) The second network uses the tightly cropped image from (A) and the LV segmentation to further segment the LV into viable and enhanced myocardium. (C) The third network is a convolutional autoencoder trained to encode (compress) and decode myocardial segmentation masks. (D) Segmentations from the training set are encoded using the third network to form a latent space. The space is modeled as a Gaussian Mixture Model (GMM) and conditional re-sampling is performed to populate the space with anatomically correct samples (black dots). Predicted segmentations are encoded and the nearest neighbors algorithm is used to return a perturbed, anatomically correct version (green dot) of the original (red dot). GMM image adapted from source [81]. Additional details are presented in Methods.

the field of view for the second network to mostly myocardium features. The ROI segmentation network (see fig. 3-3) was a U-Net with residuals (ResU-Net) of depth four [82, 68]. During the downsampling process, each of the four depth levels consisted of 2 repetitions of a block made up of a $3 \times 3$ 2-D convolution, followed by a rectified linear unit (ReLU) activation and batch normalization. After the 2 blocks, each was followed by a $2 \times 2$ max pooling layer and 20% dropout. The upsampling branch had a similar structure using $2 \times 2$ nearest neighbor upsampling and identical convolutional layers. ROI predictions were automatically cleaned up by discarding all but one connected component. Each slice’s component was chosen as the one closest to the center of mass of objects in slices located between the 20th and 80th percentiles (higher confidence) of the short-axis height. Lastly, slices with very large jumps in ROI area
close to the base (likely above the ventricle) were automatically pruned.

![Figure 3-3. Left Ventricle Region of Interest and Myocardium Segmentation Network Architecture.](image)

The left ventricle (LV) network identifies the main region of interest. The second network segments the myocardium (MYO) by differentiating between viable and non-viable tissue represented by each of the two outputs. The networks differ by the number of filters, input image size, and number of outputs as indicated. Both networks are U-Nets with residuals.

Stage 2: Left Ventricle Myocardium Segmentation Network

The second network (fig. 3-2B) used the ROI mask from the first network to differentiate LV blood pool, viable myocardium, and regions of enhancement, returning segmentations for the latter two tissue types. Using the center of mass of the slice, the LV images were centered in a $128 \times 128$ pixel image. Intensities were re-scaled based on the intensity histogram derived from each patient’s entire 2-D stack, thus preserving the intensity contrast of enhanced and non-enhanced myocardium. The median intensity (likely blood pool) was set at the midpoint of the dynamic range interval. Specifically, the following functions were applied sequentially to each 2-D

---

<table>
<thead>
<tr>
<th>F</th>
<th>LV Net.</th>
<th>MYO Net.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16N</td>
<td>192</td>
<td>128</td>
</tr>
<tr>
<td>O</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

(Conv 3x3, ReLu, Batch Norm)x2
Max Pooling 2x2, Dropout 0.2
Dropout 0.2, Upsampling 2x2
Concatenate
Conv 1x1
Upsampling 2x2, Addition
input image component-wise:

\[ I \mapsto \frac{I}{2m_{I_D}} \]
\[ I \mapsto 255 \times \frac{I - \min_D I}{\max_D I - \min_D I} \]
\[ I \mapsto \min\{255, \max\{0, I\}\} \]

where \( I \) is the image intensity, \( D \) is the effective region following cropping by the ROI segmentation network (excluding any potential zero-padding to 128 \( \times \) 128), and \( m_{I_D} \) is the median signal intensity over \( D \).

For the LV myocardium segmentation network, we implemented a modified ResU-Net structure (fig. 3-3), similar to the ROI segmentation network. It differed from the ROI network in that it used twice the number of filters at each of the 4 depth levels due to the higher complexity of the task. The network outputted two masks, one representing the entire myocardium and the other identifying only the enhanced tissue. The enhanced regions were minimally cleaned up using an automated series of morphological operations, such as erosion, opening, and closing, with parameter choices determined by a grid search on the training data only.

**Stage 3: Anatomical Autoencoder Post-Processing**

The final neural network (fig. 3-2C, D) ensured that myocardial segmentation results abided by anatomical guidelines, reducing the performance impact of ambiguous regions (e.g., apex and base), where observer ground truth variability was high primarily due to imaging artifacts. Anatomical corrections were applied on reduced-dimension versions of the myocardial segmentations. The space of low-dimensional myocardial segmentations was constructed using a convolutional autoencoder network (see fig. 3-4), which consisted of 6 strided 3 \( \times \) 3-kernel convolutional layers, 2 dense layers, and 6 transpose convolutions. Leaky ReLU activations were used after each layer. On the encoding branch, the number of channels started at 16 and doubled after each
convolutional layer to 2,048, before being collapsed to the 16-dimensional encoding vector by a dense layer.

**Figure 3-4. Autoencoder Network Architecture.** The anatomical autoencoder serves as a post-processing step and can be thought of as a separate encoder and decoder. During the training phase, it takes as input correct myocardium segmentation masks and learns to reduce their dimensionality via a convolutional encoder-decoder. The reduced training masks are used to further learn the latent space. During the prediction phase, new masks are encoded in the latent space and a nearest neighbor method is used to find the closest one in the now “densified” space. By construction this neighbor representation is “close enough” to the original image and will decode to an anatomically correct representation. See Methods for more details.

During training, the autoencoder learned how to encode ground truth myocardial segmentations into the 16-dimensional latent space and use this representation to decode into the original image, effectively resulting in a collection of 16-dimensional vectors representing anatomically correct myocardial segmentations. Next, the latent space was augmented by generating new vectors based on the existing ones. This was done by modeling the existing vectors using a Gaussian mixture model and performing rejection sampling, where the rejection criteria encapsulated the anatomical correctness using a collection of morphological checks (see Appendix II).

Expanding on the work in Painchaud et al. [83], we developed a binary function $\delta(\cdot)$ which uses different morphological operations to determine if a myocardium mask is anatomically correct. This function checks for convexity defects, holes in myocardium, circularity thresholds, number of objects, and myocardial wall thickness.
The convolutional autoencoder was trained to reproduce myocardial segmentations after encoding them to a \( d \)-dimensional vector via a map \( \phi \) (see fig. 3-4), which is approximately invertible: let the decoding function be \( \phi^{-1} \). Given the limited data, the low-dimensional vectors of the masks populating the latent space of the autoencoder may not be sufficient to capture the diverse geometries of valid segmentations. Therefore, the latent space is augmented with a large number of \( d \)-dimensional vectors, \( z \), such that when decoded by the network, \( \delta(\phi^{-1}(z)) = 1 \).

We fit a Gaussian mixture model (GMM) with \( k \) components to the training \( d \)-dimensional vectors in the latent space. We estimated \( k = 5 \) and \( d = 16 \) using the negative log likelihood (NLL) and adjusted Akaike information criterion (AIC) [84] by cross-validation on the training set. In order to avoid penalizing high dimensional fits with many small singular values in the covariance matrix, we adjusted the standard AIC by scaling the number of parameters by the effective rank \( \text{Tr}(\Sigma)/\sigma_{\text{max}}(\Sigma) \), where \( \text{Tr} \) is the trace, \( \sigma_{\text{max}} \) is the spectral norm, and \( \Sigma \) is the covariance matrix of a GMM component (fig. 3-5). When sampling from the new distribution, we rejected a vector if, once decoded, the resulting mask does not pass the anatomical check \( \delta \). Once trained, the autoencoder’s latent space is populated using this re-sampling scheme with vectors which are ensured to decode to anatomically correct masks. When predicting, a new — potentially incorrect — mask, \( \hat{I} \), such mask is first encoded (fig. 3-2C) in the latent space to a vector \( \hat{z} = \phi(\hat{I}) \in \mathbb{R}^d \). If \( \delta(\phi^{-1}(\hat{z})) \neq 1 \), that is, \( \hat{I} \) does not encode and decode to an anatomically correct image, a nearest-neighbor algorithm is used to find the closest match \( \hat{z}_{NN} \) in the latent space given by

\[
\hat{z}_{NN} := \arg \min_{z \text{ s.t. } \delta(\phi^{-1}(z)) = 1} \| z - \hat{z} \|^2,
\]

where \( z \) varies over the constructed couples that satisfy anatomical constraints. Lastly, we define the final, anatomically correct segmentation as \( \hat{I}^* = \phi^{-1}(\hat{z}^*) \), where

\[
\hat{z}^* = \hat{z} + \alpha^*(\hat{z} - \hat{z}_{NN})
\]
and $\alpha^*$ is the smallest $\alpha$ in $[0, 1]$ such that

$$\delta[\phi^{-1}(\hat{z} + \alpha(\hat{z} - \hat{z}_N))] = 1$$

as illustrated in fig. 3-2D. This increased the size of the latent space by sampling an additional 10,000 points. Therefore, though this process, we guarantee anatomical correctness of the predicted myocardium mask.

Finally, the 2-D myocardial segmentations were reconstructed to volumes and additional automatic volumetric checks were applied to remove segmentations from images located below the apex or above the base of the LV. We compared ratios of myocardium to blood pool areas of each slice to identify the longest sub-sequence of slices in the stack. The threshold used to determine whether to include a slice in the sub-sequence was approximately a 40% maximum decrease in LV area. Segmented volumes were truncated at the index $i = \max(i_M, \min(i_C + 1, i_D))$, where $i_M$ refers to the final index in the sub-sequence; $i_C$ represents the index of the first “C”-shaped slice (a myocardial segmentation shape that occurs at the boundary of the ventricle.
and the atrium in the basal region); and \( i_D \) represents the index of a large deviation (drop to 60% or increase of 60%) in LV area between successive slices. This check allowed incorporation of at most one “C”-shaped slice and excluded slices above the base with no true region of interest. The numerical values for the thresholds were determined by ensuring no more than 5% of the ground truth segmented slices would be discarded. Final predicted myocardial segmentations of patient scans therefore passed both per-slice and per-volume anatomical constraints.

**Training and Evaluation**

The training data set consisted of 2,484 images from two sources: 1,124 2-D LGE-CMR slices from 75% of available patients and all 1,360 “LGE-like” images. The test set contained only LGE-CMR images from the remaining 25% of patients (269 2-D images). For the myocardium segmentation network, only LGE-CMR scans with enhancement segmentation ground truth were used (roughly 80% of the train and test sets). The autoencoder used ground truth myocardial segmentations from all the available training data. No early stopping or other methods that learn from the validation set were used in training.

To prevent cine-derived “LGE-like” images from dominating the training set, they were weighed less in the loss function. The loss function used was an equally weighted combination of the balanced cross-entropy loss:

\[
l_1(p, \hat{p}) = -\beta p \log(\hat{p}) + (1 - \beta)(1 - p) \log(1 - \hat{p})
\]

and the Tversky loss [85]

\[
l_2(p, \hat{p}) = 1 - \frac{2TP(p, \hat{p})}{2TP(p, \hat{p}) + \beta FP(p, \hat{p}) + (1 - \beta) FN(p, \hat{p})},
\]

where \( p \) and \( \hat{p} \) are pixel ground truth and predicted values, T/F P/N are true/false positive/negatives, and \( \beta \) is a weight on the false positives, which was modulated up to \( \beta = 0.6 \) in the first network to avoid over-cropping and down to \( \beta = 0.4 \) in the
second to limit outliers. The final loss combined per-pixel mean loss ($l_1$) and per-image ($l_2$) loss in equal proportions to incorporate both local and holistic performance. All networks used the Adam optimizer \[33\] with learning rate of $10^{-3}$ and trained on NVIDIA Titan RTX graphics processing units using \texttt{keras} [2] and \texttt{tensorflow} [4].

We evaluated ACSNet’s segmentation performance (Table 3-I) using balanced accuracy, Dice coefficient, and Hausdorff distance (HD) \[86\] as metrics (see Appendix I). Values were computed per section of the heart (apex, mid-ventricle, base) and for the total heart. Sections of the heart were determined by equipartitioning the short axis distance between the first and last slice.

Additionally, we evaluated LV ROI (myocardium and blood pool) volume, myocardium volume, enhancement region volume, and core scar region volume derived from the segmentations (Table 3-II). Volumes were calculated by summing voxel volumes and using nearest-neighbor interpolation between slices. To quantify core scar, the enhanced (scar/fibrosis) region segmented by the network was used to extract the dense core scar region using a modified version of the full width at half maximum (FWHM) \[87, 88\] algorithm. The remote non-enhanced myocardium intensity used by the FWHM algorithm was automatically determined as the median intensity value outside the predicted enhancement region. Differences between ground truth and predictions were reported as the mean absolute error (MAE) normalized relative to the ground truth value.

\section*{Statistical Methods}

All data analysis in this manuscript was performed using Python 3.4 and open source packages. All results presented without a qualifier represent averages over slices or patients from the 25\% of the contrast-enhanced data reserved for testing using a random split. Prediction error was estimated using approximately normal confidence intervals for large $n$ (\textit{e.g.}, number of slices) and minimum/maximum ranges for small
n (e.g., number volumes). Statistically significant difference testing was assessed using Welch’s t-test using the package scipy.

Results

Segmentation Performance

ACSNet segmentations were evaluated for overlap with ground truth data using Dice score and for geometric outliers and extremes using the highly sensitive Hausdorff distance (HD). Training on both “LGE-like” cine and LGE-CMR slices resulted in Dice scores of 0.96 when identifying the LV ROI (first sub-network) and 0.91 when differentiating the myocardium from the blood pool (second sub-network). Segmentations predicted from our test set of LGE-CMR scans achieved Dice scores of 0.93 and 0.79 on the two sub-networks respectively. With anatomical post-processing by the third sub-network, ACSNet maintained a Dice score of 0.79±0.02 (93% balanced accuracy) for LV myocardial segmentations. Figure 3-6 illustrates the consistency of ACSNet results across the three regions of the LV (apex, middle, and base) through histograms of per-slice Dice scores. Dice scores are shown for the LV ROI (fig. 3-6(A-C)) and LV myocardial segmentations (fig. 3-6(D-F)). The average Dice scores for each region are 0.94 (A) and 0.80 (D) for basal slices, 0.95 (B) and 0.82 (E) for middle slices, and 0.92 (C) and 0.75 (F) for apical slices. Thus, each region’s average Dice score fell within 4% of the overall average. The HD was 6.72 ± 0.58 millimeters (mm) between manual and ACSNet segmentations. Detailed results of balanced accuracy, Dice score, and HD are shown for LV and myocardial segmentations for the three regions of the ventricle in table 3-1(LV ROI, MYO).

Table 3-III presents a comparison of Dice scores and HD for previous LV myocardial segmentation methods, showing that ACSNet achieved the lowest HD of previously recorded LGE-CMR myocardium segmentation methods. The Dice score is similar
Figure 3-6. **Left Ventricle Region of Interest and Myocardium Segmentation Results by Region.** Histograms of per-slice Sørensen-Dice scores are shown for three regions of the heart (rows from top to bottom: basal, middle, and apical). Columns represent the left ventricle region of interest (LV ROI) segmentation (A-C) and myocardium (MYO) segmentation (D-F). The averages are shown as solid vertical lines, and the dotted lines represent the 5th and 95th percentiles.

ACSNet improved upon both the inter-observer Dice score of 0.76 as well as the inter-observer HD (10.6 ± 4.65 mm endocardial HD and 12.5 ± 5.38 mm epicardial HD) recorded in the Multi-Sequence Cardiac MRI Segmentation Challenge [89, 90].
Table 3-I. ACSNet LGE-CMR Segmentation Performance. Balanced accuracy (BA), Sørensen-Dice coefficient (Dice), and Hausdorff distance (HD) are shown for 4 regions of interest segmented by ACSNet: whole left ventricle region of interest (LV ROI), myocardial tissue (MYO), area of enhancement (Enhancement Region), and scar tissue (Core Scar Region). BA is expressed in percentage terms, Dice is adimensional, and HD is in millimeters. All numbers are averages ± 95% confidence interval size over apex/middle/base/total slices of all patients in the test set.

Figure 3-7 illustrates ACSNet’s performance in terms of scar segmentation. The first row shows the original scan, the middle row presents the ground truth scar and GZ segmentations, and the bottom row shows the predicted segmentations. Results for patients 1-3 are representative examples of scar and GZ segmentations. Patient 4 was included to show an example of an outlier for which GZ segmentation has low accuracy; this, however, does not affect the scar segmentation. Segmentation of the enhancement region showed a balanced accuracy of 70%. As evident in fig. 3-7, the balanced accuracy after thresholding increased to 75% for the core scar. Balanced accuracy, Dice score, and HD for all enhancement and core scar for the apex, middle, and base.
of the LV are shown in table 3-1 (Enhancement Region, Core Scar Region).

**Figure 3-7. Scar Segmentation Results.** Segmentations of enhancement regions from the myocardium segmentation network represent gray zone (yellow) and scar (red). We illustrate the original scan (first row), the ground truth scar and gray zone segmentations (middle row), and the predicted segmentations (bottom row) using ACSNet.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Lower</th>
<th>Middle</th>
<th>Upper</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LV ROI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT (cc)</td>
<td>226 (186 – 259)</td>
<td>307 (280 – 327)</td>
<td>405 (334 – 573)</td>
<td>312 (186 – 573)</td>
</tr>
<tr>
<td>Norm. MAE (%)</td>
<td>10.3 (4.8 – 18.8)</td>
<td>4.5 (0.6 – 8.0)</td>
<td>4.4 (1.3 – 10.3)</td>
<td>6.3 (0.6 – 18.8)</td>
</tr>
<tr>
<td><strong>MYO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT (cc)</td>
<td>121 (85 – 159)</td>
<td>171 (110 – 215)</td>
<td>180 (114 – 274)</td>
<td>158 (85 – 274)</td>
</tr>
<tr>
<td>Pred (cc)</td>
<td>144 (109 – 186)</td>
<td>187 (144 – 226)</td>
<td>217 (159 – 351)</td>
<td>183 (109 – 351)</td>
</tr>
<tr>
<td>Norm. MAE (%)</td>
<td>24.1 (12.6 – 41.1)</td>
<td>12.7 (1.8 – 30.8)</td>
<td>24.7 (2.2 – 53.7)</td>
<td>20.1 (1.8 – 53.7)</td>
</tr>
<tr>
<td><strong>Enhancement Region</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT (cc)</td>
<td>27 (15 – 46)</td>
<td>24 (3 – 39)</td>
<td>30 (7 – 47)</td>
<td>27 (3 – 47)</td>
</tr>
<tr>
<td>Pred (cc)</td>
<td>21 (7 – 43)</td>
<td>19 (0 – 33)</td>
<td>26 (3 – 38)</td>
<td>22 (0 – 43)</td>
</tr>
<tr>
<td>Norm. MAE (%)</td>
<td>26.6 (5.9 – 53.4)</td>
<td>31.1 (3.8 – 100.0)</td>
<td>45.4 (15.9 – 87.5)</td>
<td>34.2 (3.8 – 100.0)</td>
</tr>
<tr>
<td><strong>Core Scar Region</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT (cc)</td>
<td>13 (6 – 21)</td>
<td>11 (1 – 19)</td>
<td>17 (4 – 30)</td>
<td>13 (1 – 30)</td>
</tr>
<tr>
<td>Pred (cc)</td>
<td>11 (5 – 20)</td>
<td>9 (0 – 17)</td>
<td>14 (2 – 22)</td>
<td>12 (0 – 22)</td>
</tr>
<tr>
<td>Norm. MAE (%)</td>
<td>14.4 (1.2 – 41.2)</td>
<td>42.5 (18.2 – 100.0)</td>
<td>42.7 (32.1 – 72.0)</td>
<td>33.7 (1.2 – 100.0)</td>
</tr>
</tbody>
</table>

Table 3-II: ACSNet LGE-CMR Clinical Feature Performance. Ground truth (GT) and predicted (Pred.) volumes and mean absolute error normalized by GT volume (Norm. MAE) together with ranges (parentheses) are shown for 4 regions of interest segmented by ACSNet: whole left ventricle (LV), myocardial tissue (MYO), area of enhancement (Enhancement Region), and scar tissue (Core Scar Region). GT and Pred. are expressed in cubic centimeters and Norm. MAE in percentage terms. Numbers represent averages across all patients in the test set (Total) and patients grouped by GT LV volume tertile (Lower/ Middle/Upper).
<table>
<thead>
<tr>
<th>Method</th>
<th>MYO Dice Score</th>
<th>MYO Hausdorff Distance (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSNet</td>
<td>0.79 ± 0.02</td>
<td>6.70 ± 0.53</td>
</tr>
<tr>
<td>Interobserver [89, 90]</td>
<td>0.76 ± 0.08</td>
<td>12.50 ± 5.38</td>
</tr>
<tr>
<td>Zabihollahy et al.[73]</td>
<td>0.85 ± 0.03</td>
<td>19.21 ± 4.74</td>
</tr>
<tr>
<td>Yue et al.[69]</td>
<td>0.76 ± 0.23</td>
<td>11.04 ± 5.82</td>
</tr>
<tr>
<td>Roth et al.[70]</td>
<td>0.78</td>
<td>16.30</td>
</tr>
<tr>
<td>Mean Result of MS-CMRSeg MICCAI Challenge [91]</td>
<td>0.77 ± 0.10</td>
<td>18.06 ± 12.18</td>
</tr>
<tr>
<td>Chen, Ouyang et al.[71]</td>
<td>0.83 ± 0.04</td>
<td>12.45 ± 3.14</td>
</tr>
</tbody>
</table>

Table 3-III. Comparison of LGE-CMR segmentation results for the LV myocardium. All entries were rounded from the provided values to the nearest tenths place. Note: These sources use different datasets; Data for Interobserver [89, 90], Yue et al.[69], Roth et al.[70], and Chen, Ouyang et al.[71] are based on the 2019 CMRSeg MICCAI challenge [91] consisting of 2-D LGE-CMR and corresponding steady-state free precision (bSSFP) from 45 patients, various subsets of whom were used as test sets. Zabihollahy et al.[73] used three orthogonal views of 34 subjects with 3-D LGE-CMR scans.

Feature Extraction Outcomes

ACSNet was used to seamlessly derive clinical features such as scar burden and LV volume. Our results demonstrate no statistically significant difference between features computed using automatic versus manual (expert-level) segmentations. LV volume was calculated as the sum of myocardium and blood pool volumes from both predicted and manual segmentations. Error for LV and scar volumes is calculated as the absolute error normalized by each respective volume. Figure 3-8(A) shows the normalized error of each LV volume in the test set. The MAE was 6.3% with a P-value of 0.71 for a difference-in-means test. Table 3-II(LV) presents the MAE and volume in cubic centimeters (cc) of ventricles grouped by size of the LV. This table additionally provides details on LV myocardium, enhancement, and core scar volume calculations. Scar volume quantification has an MAE of 33.7% and P-value of 0.46. Figure 3-8(B) shows a plot of each patient’s error for scar volume calculation. Scar burden, calculated as the mean scar-to-myocardium volume fraction, differed by 2% when comparing automatic and manual segmentations. All segmentations passed both slice- and volume-level anatomical checks.
Figure 3-8. Scar and LV Volume Calculation Results. LV (A) and scar (B) volume error is computed as the absolute error normalized by each respective volume. Each point represents the error in LV volume of a single segmented patient scan. The solid black line shows the mean.

Discussion

In this study we present a DL approach for automatic and anatomically accurate segmentation of myocardium and scar/fibrosis on LGE-CMR images and for extraction of anatomical features, such as scar burden and ventricular volume. The complex learning process involves three sub-networks, each having distinct tasks: the first reduces class imbalance between the ROI and background, the second delineates the endocardium and epicardium, and the third ensures anatomical correctness for both slices and volumes. We show that ACSNet outperforms inter-expert (i.e., manual) scores and performs well on inputs with various scar distribution patterns acquired from numerous imaging centers and MR machines. Our technology is seamlessly employed to extract clinical anatomical covariate data, potentially enhancing the prognostic utility of LGE-CMR.

Advantages of ACSNet

ACSNet fully automates the segmentation of LGE-CMR images. The significant amount of manual labor and the inter-observer variability associated with this task have posed a barrier to implementing LGE-CMR image analysis as part of the patient assessment and prognostication process. For instance, scar burden and LV volume
computed from myocardial and scar/fibrosis segmentations have been associated with risk of sudden cardiac death [92] but are not used in practice to guide primary prevention. ACSNet can make highly accurate segmentations available within seconds from a set of raw medical images, making it possible to incorporate LGE-CMR image analysis in clinical decision-making, thus increasing the tractability of LGE-CMR in sudden cardiac death risk stratification.

ACSNet achieves high performance despite the complexity of LGE-CMR images. Contouring of LGE-CMR images is complicated by the presence of both low (viable) and high (scar/fibrosis) signal intensity myocardium regions. As a result, manual segmentations are highly variable across experts, leading to poorly estimated clinical features. The same reason leads computer-aided segmentation algorithms to struggle with distinguishing myocardium from blood pool. ACSNet’s results demonstrate its ability to generate reliable segmentations when ground truth data is noisy due to the high inter-observer variability. The network maintains consistently high performance across all regions of the heart, which here is prioritized over a high average Dice score with poor-performing outlier slices. Despite ACSNet’s success with whole-ventricle segmentation, an outlier may occur with a poor-performing GZ segmentation (see fig. 3-7, Patient 4). However, despite the potential of occasional such outliers, ACSNet maintains a low error in scar prediction upon thresholding at the FWHM intensity value. This low error ensures the accuracy of clinically important features calculated from ACSNet scar segmentations.

In this study, we use a novel method to ensure anatomical accuracy of segmentations: we integrate, in the DL approach, an additional deep neural network (ACSNet’s third sub-network) encompassing a number of per-slice and per-volume morphological checks. Our refined model of the latent space distribution allows for complex anatomical segmentations such as “C”-shaped myocardium, should such shapes arise in basal slices. Moreover, ACSNet uses volumetric checks that standardize the selection of
apical and basal (beginning and end) slices in our diverse set of data, a time-consuming and often variable process when performed manually. These checks also establish consistency and reliability in the calculation of clinical features (e.g., LV volume and scar burden).

Cine data is widely available with ground truth labels for algorithmic training; however, publicly available LGE-CMR data is scarce. Importantly, ACSNet performs well despite data scarcity due to the innovative style transfer process to augment the training data presented here. This process generates pseudo-enhancement for non-enhanced cine using a low-cost cine-to-LGE conversion algorithm. The method tripled the available dataset and expanded ACSNet’s ability to be successfully used across patient cohorts, MR scanners (Siemens®, Philips®, and General Electric®), and health centers.

By training our networks with both LGE and “LGE-like” cine CMR images from a broad range of cohorts, we have created a technology potentially capable of fully automated segmentation of any short-axis cardiac images. Because we use style-transferred cine images in our training, ACSNet is expected to segment cine scans with high accuracy. It is also expected to be generalizable to computed tomography images, which, like CMR, display a high-intensity blood pool and low-intensity myocardium. Finally, ACSNet performance was validated on ICM patient data, though ACSNet could be easily extended to non-ICM patient scans as well.

Further, ACSNet is a prime technology to employ in basic science research. Research on heart disease in animal models may also benefit from ACSNet in image analysis. Furthermore, personalized computational heart models often use segmented images to conduct simulations that identify arrhythmogenic pathways and arrhythmia dynamics [63, 64, 65] or the targets for ablation therapy [93, 94].
Comparison to Other Work

Segmentation algorithms for the LV myocardium have commonly focused on cine CMR images. Despite promising advances, cine segmentation algorithms still often require manual steps. For example, the method by Zheng et al. [68] requires a preprocessing step to discard apical and basal slices and a manual curation of “difficult cases”. An attempt by Bello et al. [67] at segmenting cine images relies on ground truth landmark annotations to prevent anatomically inconsistent outliers. The persistent limitations in cine segmentation have demonstrated that LGE-CMR require a tailored segmentation algorithm rather than a reimplemention of methods developed for cine scans.

Some DL methods have been proposed specifically for LGE-CMR myocardial or scar segmentation, however, these solutions also have a number of limitations. The approach by Campello et al. [72] for segmenting the myocardium in LGE-CMR images attempted to address LGE-CMR data scarcity by using a costly DL cine-to-LGE style transfer approach. However, in the process, the style-transferred cine images lost the salient aspect of LGE-CMR, the scar/fibrosis features. A recent attempt by Zabihollahy et al. [73] at myocardial and scar/fibrosis segmentation on 3-D LGE-CMR resulted in artifacts, such as disjoint pieces of the myocardium, despite the benefit of a 10-fold increase in the number of slices per patient furnished by the 3-D acquisition. The 2019 CMRSeg MICCAI challenge for myocardial segmentation [91, 69, 70, 71] and a study focused on scar segmentation [76] both required cine and LGE-CMR scans for each patient. Furthermore, Fahmy et al. [76] exclusively utilized images of patients with hypertrophic cardiomyopathy with unknown generalizability to other cardiomyopathies. An attempt by Moccia et al. [74] at predicting enhancement segmentations required manually segmented ground truth myocardium as an additional network input; this requirement limited their dataset to only 30 patients, all from a single center.
A few recent methods have proposed post-processing steps to improve the anatomical accuracy of myocardial segmentations from cine images \cite{83, 95}. Although these algorithms smooth out resulting segmentations, they use generic techniques unable to capture nuances of heart anatomy \cite{95}; they require an already highly accurate segmentation as input to function well \cite{83}; or they do not incorporate 3-D constraints \cite{83}.

**Study Limitations**

As with many DL algorithms, ACSNet could benefit from a larger cohort with more diverse image data such as scar distribution patterns. While the pseudo-enhancement generated for our synthetic data was randomized, and not based on common scar distributions, nevertheless ACSNet succeeded in identifying scar, and the addition of pseudo-enhancement improved network performance. Additionally, we did not perform a broad hyperparameter sweep when developing the structure of our networks. However, manual exploration of different hyperparameter implementations resulted in similar Dice scores for myocardium segmentation, which indicates the robustness of the core ACSNet model.

**Conclusion**

We demonstrated an efficient, cost-effective DL solution for automatic expert-level segmentation of LGE-CMR images and seamless, accurate derivation of scar burden and LV volume. ACSNet segmentations abide by anatomical constraints and have been trained on scans from heterogeneous cohorts, thus offering the potential for much broader application across imaging modalities and patient pathologies.
Chapter 4

Deep Learning Survival Analysis For Arrhythmia Prediction

Introduction

Sudden cardiac death (SCD) continues to be a leading cause of mortality worldwide, with an incidence of 50 to 100 per 100,000 in the general population in Europe and North America [13, 14], and accounts for 15–20% of all deaths. The majority of arrhythmic SCD (SCDA) events occur in patients with coronary heart disease [15]. While implantable cardioverter devices (ICD) effectively prevent SCD due to ventricular arrhythmias, current clinical criteria for ICD candidacy — i.e., left ventricular ejection fraction (LVEF) < 30–35% [16] — only capture a mere 20% all SCDA [17], highlighting the critical need to develop personalized, accurate, and cost-effective arrhythmia risk assessment tools to mitigate this enormous public health and economic burden.

Many studies have identified risk factors for SCDA and numerous risk stratification approaches have attempted to transcend LVEF [96, 97]. However, limitations in these approaches have been barriers to their clinical implementation. Prior attempts have broadly stratified populations based on subgroup risk, failing to customize predictions to patients’ unique clinical features [98]. SCDA risk has been typically assessed at pre-defined finite time points, ignoring the likely patient-specific time-evolution of the disease [99]. Additionally, in previous work confidence estimates for predictions
have been “one-size-fits-all”, varying only by risk subgroup, thus preventing the identification of low confidence, potentially highly erroneous prediction outliers [100]. Moreover, few prior studies have validated their results externally or comprehensively compared model performance to standard approaches. A robust, generalizable SCDA risk stratifier with the ability to predict individualized, patient-specific risk trajectories and confidence estimates could significantly enhance clinical decision-making. Finally, although arrhythmia arises, mechanistically, from the heterogeneous scar distribution in the disease-remodeled heart, learning the features of that distribution has not been explored for risk analysis. Image-derived mechanistic computational models of cardiac electrical function that incorporate scar distribution have proven successful in predicting arrhythmia risk [101], however, they remain exceedingly computationally intensive and, therefore impractical as a first stage screening tool in a broad population. Using raw contrast-enhanced (LGE) cardiac images that visualize scar distribution in a DL framework which additionally draws on standard clinical covariates, would overcome all these limitations and lead to accurate patient-specific SCDA probabilities in fractions of a second.

Here, we present a deep learning (DL) technology for prediction of SCDA risk in patients with ischemic heart disease. Our approach, which we term Survival Study of Cardiac Arrhythmia Risk (SSCAR), embeds, within a survival model, neural networks to estimate individual patient times to SCDA ($T_{SCDA}$). The neural networks learn from raw clinical imaging data that visualize post-infarct scar distribution as well as from clinical covariates. The predicted patient-specific survival curves offer accurate SCDA probabilities at all times up to 10 years. The performance and high generalizability of the approach are demonstrated by testing on an external cohort, following internal cross-validation. Our technology represents a fundamental change in the approach to arrhythmia risk assessment, as SSCAR uses the data to directly estimate uncertainty in its predictions. SSCAR thus has the potential to significantly
shape clinical decision-making regarding arrhythmia risk, offering not a simple “at risk/not at risk” prediction, but instead, an estimate of the time to SCDA together with a sense of “how certain” the model is about each predicted $T_{SCDA}$.

Methods

Survival Model

Statistical Fit

For each patient $i$, let $T_i^{(1)}$ be defined as the time to arrhythmic death and let $T_i^{(2)}$ be the time to any event that represents a competing death (i.e., non-arrhythmic death). Further, let $C_i$ be the right-censoring time (e.g., loss of follow-up). The identifiable outcome data was the pair $(X_i, \Delta_i)$, where $\Delta_i = (\Delta_i^{(1)}, \Delta_i^{(2)})$ and

$$X_i = \min\{T_i^{(1)}, T_i^{(2)}, C_i\}$$

$$\Delta_i^{(k)} = \begin{cases} 1 & \text{if } T_i^{(k)} = X_i \\ 0 & \text{otherwise,} \end{cases}$$

for $k = 1, 2$, indicating the two disjoint events. We assume that patients have independent times and the event times are independent of the censoring times. In what follows, for ease of notation, we suppress the dependency on the parameter(s) we wish to infer and re-introduce them after this discussion. We define the cause-specific hazard function $\lambda_i^{(k)}(t)$ as

$$\lambda_i^{(k)}(t) = \lim_{h \to 0} \mathbb{P}\left\{ t < X_i \leq t + h, K = k | X_i \geq t \right\}$$

and note that the survival function can be broken down as

$$S_i(t) = \exp \left\{ - \int_0^t \lambda_i(u) du \right\}$$

$$= \exp \left\{ - \int_0^t \left[ \lambda_i^{(1)}(u) + \lambda_i^{(2)}(u) \right] du \right\}$$

$$= \exp \left\{ - \int_0^t \lambda_i^{(1)}(u) du \right\} \exp \left\{ - \int_0^t \lambda_i^{(2)}(u) du \right\}$$

$$= S_i^{(1)}(t) S_i^{(2)}(t),$$
where $S_i^{(1)}$ and $S_i^{(2)}$ are patient- and cause-specific (pseudo)survival function. This notation allows us to write the likelihood function as \[49\]:

$$
L = \prod_{i=1}^{n} \lambda_i^{(1)}(x_i)^{\delta_i^{(1)}} \lambda_i^{(2)}(x_i)^{\delta_i^{(2)}} S_i^{(1)}(x_i) S_i^{(2)}(x_i)
$$

We now make the main assumption regarding the distribution of the times to event. We assume that the cause-specific hazard rate is that from a log-logistic distribution with location parameter $\mu_i$ and scale parameter $\sigma_i$, such that

$$
S_i^{(1)}(t; \mu_i, \sigma_i) = \frac{1}{1 + \exp[(\log t - \mu_i)/\sigma_i]}.
$$

The patient-specific parameters $\mu_i$ and $\sigma_i$ were modeled as outputs of neural networks applied to LGE-CMR images and clinical covariates. This results in the profile log-likelihood

$$
- \log L_1 = - \sum_i \log \frac{x_i - \mu_i}{\sigma_i} + \delta_i^{(1)} \log \sigma_i + (1 + \delta_i^{(1)}) \log \left[1 + \exp\left(\frac{\log x_i - \mu_i}{\sigma_i}\right)\right],
$$

where $x_i$ is the observed time and $\delta_i^{(1)}$ the censoring status.

**Performance Metrics**

The all-time performance of the models was evaluated using two measures. The first was Harrell’s c-index [102] with the patient-specific $\mu_i$’s as the risk scores ($\exp(\mu_i)$ is the mode of the log-logistic distribution) to gauge the model’s risk discrimination ability. The second was the integrated Brier score [103], which is defined as the time-average of mean squared error between true 0/1 outcome and predicted outcome probability and gauges probability calibration. Both measures were adjusted for censoring, corrected by weighing with the inverse probability of censoring, and calculated for data prior to a given cut-off time $\tau$ [104]; if unspecified, $\tau = 10$ years, corresponding with the maximum event time in the data set. Metrics derived from the confusion matrix (e.g., precision and recall) were computed at several time points ($\tau = 2, 3 \ldots$ years). Probability thresholds at these times were selected by maximizing F-score (precision,
recall, F-score) or Youden’s J statistic (sensitivity, specificity, balanced accuracy) on the training data. For additional details on performance metrics, see Appendix I.

**Neural Network Architecture**

SSCAR is a supervised survival analysis regression model composed of two sub-networks, each operating on different input types (see Fig. 4-1): a “convolutional” sub-network which takes the LGE-CMR images as inputs, and a “dense” sub-network which uses the covariate data. Feature extraction in the convolutional sub-network from the LGE–CMR images was achieved by a 3-dimensional convolutional encoder-decoder model. The encoder used a sequence of 3-D convolutions and pooling layers, followed by one dense layer to encode the original 3-D volume into a lower-dimensional vector. Nonlinear activation functions and dropout layers were added before each downsampling step. The encoding was further used for two purposes: survival and reconstruction. For the survival branch, the encoding was first stratified into one of \( r \) (learned) risk categories (see Table 4-I) and then fed to a 2-unit dense layer to predict — for each patient — a set of 2 parameters, location \( \mu \) and scale \( \sigma \), which fully characterized the probability distribution of the patient’s log-time to SCDA (see Statistical Fit), followed by a bespoke activation function. This activation function clipped \( \ln \mu \) on \([-3, 3]\) and clipped \( \sigma \) from below at \( \sigma_{\text{min}} \), where \( \sigma_{\text{min}} \) was found such that the difference between the 95th and 5th percentiles of the predicted \( T_{SCDA} \) distribution was no less than a month. This survival activation function effectively restricted the “signal-to-noise” ratio \( \mu/\sigma \) (see Appendix II for details on implementation). For the purpose of reconstruction, the encoding was decoded via a sequence of transposed convolutions to re-create the original volume. Feature extraction from the clinical covariate data was performed using a sequence of densely connected layers, followed by a dropout layer to prevent overfitting. The resulting tensor used a similar path to the one followed by the convolutional encoding to
eventually map to the 2 survival parameters. Finally, once the two sub-networks were trained, they were frozen and joined using a linear combination layer to ensemble the survival predictions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Convolutional [Choice]</th>
<th>Dense [Choice]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs (maximum)</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Learning rate (initial)</td>
<td>.01</td>
<td>.01</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Steps per epoch</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Activation function</td>
<td>ReLU</td>
<td>ReLU</td>
</tr>
<tr>
<td>Convolutional kernel size</td>
<td>3 × 3 × 3</td>
<td>N/A</td>
</tr>
<tr>
<td>Network depth</td>
<td>{2, 4} [2]</td>
<td>{1, 2, 3, 4, 5} [3]</td>
</tr>
<tr>
<td>Risk categories</td>
<td>{1, 2, 3} [2]</td>
<td>{1, 2, 3} [1]</td>
</tr>
<tr>
<td>Dropout</td>
<td>Uniform(0, 0.5) [0.04]</td>
<td>Uniform(0, 0.5) [0.22]</td>
</tr>
<tr>
<td>l1 (bias and kernel) regularization</td>
<td>LogUniform(10⁻⁷, 10⁻³)</td>
<td>4.62 × 10⁻³</td>
</tr>
<tr>
<td>l2 (bias and kernel) regularization</td>
<td>LogUniform(10⁻⁷, 10⁻³)</td>
<td>3.25 × 10⁻⁶</td>
</tr>
<tr>
<td>Convolutions per sampling level</td>
<td>{1, 2} [1]</td>
<td>N/A</td>
</tr>
<tr>
<td>Convolutional filters (initial)</td>
<td>{8, 12, 16, 20, 24} [24]</td>
<td>N/A</td>
</tr>
<tr>
<td>Reconstruction loss weight</td>
<td>Uniform(0, 10) [.06]</td>
<td>N/A</td>
</tr>
<tr>
<td>Latent representation dimension</td>
<td>Uniform(10, 30) [22]</td>
<td>N/A</td>
</tr>
<tr>
<td>Units per dense layer</td>
<td>N/A</td>
<td>Uniform(4, 24) [8]</td>
</tr>
</tbody>
</table>

**Table 4-I. Hyperparameter Tuning.** The hyperparameters used to define network architectures and training guidelines (left column) were optimized by randomly sampling using the Parzen windows algorithm from the search spaces defined for the convolutional (middle column) and dense sub-networks (right column). Braces denote equally probable sampling from the set. “Uniform” refers to uniformly distributed either continuously (parentheses) or discretely (braces). “LogUniform” refers to uniformly distributed exponent. No qualification means the hyperparameter was fixed. Network depth is the number of convolution/downsample steps for the convolutional sub-network and the number of dense layers for the dense sub-network. Risk categories refers to the number of strata used to divide before the final layer (see details in Methods). The latent representation dimension is the number of units of the dense layer immediately following the last convolution/downsample step. The hyperparameter was stopped after 100 and 300 iterations for the convolutional and dense model, respectively. Choices for the final model are displayed in bold in square brackets.

The predicted survival parameters (location and scale) aimed to minimize the aforementioned negative log likelihood function for the log-logistic distribution, accounting for censoring in the data and class imbalance. The reconstructed output of the convolutional sub-network minimized the mean squared error (MSE) to the original input. Its contribution to total loss was learned to provide regularization to the imaging features extracted, ensuring the survival fit relied on features able to reconstruct the original image. Both stochastic gradient descent (SGD) and Adam[33] optimizers
Figure 4-1. Schematic Overview of SSCAR. Top panel (yellow) shows patient data used in this study. SSCAR uses contrast-enhanced (LGE) cardiac magnetic resonance (CMR) images with the left ventricle automatically segmented (left inset) and clinical covariates (right inset, see Table 4-II for a complete list) as inputs to the two sub-networks (left and right pathways). Labels associated with each patient (SCDA Events, middle inset, dot-dashed contour) — consisting of the observed time to event, and an indicator whether the event was sudden cardiac death from arrhythmia (SCDA) or non-SCDA — are used as targets during training only. LGE-CMR data is taken as input by a 3-D convolutional neural network constructed using an encoder-decoder architecture (red panel, left). Clinical covariates are fed to a dense neural network (green panel, right). The sub-networks are trained to estimate two parameters (location, $\mu$ and scale, $\sigma$) specific to each patient, which fully characterize the probability distribution of the patient-specific time to SCDA (top blue panel; the time to SCDA is modeled as probabilistic, assumed to follow a log-logistic distribution). During training (dot-dashed arrows and white middle panel), the neural network weights are optimized via a maximum likelihood process, in which a probability distribution is sought (blue double-headed arrow in middle white panel) to best match the observed survival data (yellow “x”’s in middle white panel). Finally, the optimized probability function is used on test LGE-CMR images and covariates to predict patient individualized survival curves (blue bottom panel).
were used. All code was developed in Python using Keras[2] and Tensorflow[4]. Each train/evaluate fold took 3–5 minutes on an NVIDIA Titan RTX.

Training and Testing

The entire model development and internal validation were performed using the LVSP-SCD cohort. Following a hyperparameter tuning step, the best model architecture was then used on the entire internal validation set to find the best neural network weights. As the ensembling step was hyperparameter-free, it did not use hyperparameter tuning.

Hyperparameter tuning. A hyperparameter search was performed using the set of parameter values described in Table 4-I, given the vast number of hyperparameter configurations available to define the model architectures. The package hyperopt [39] was used to sample parameter configurations from the search space using the Parzen window algorithm to minimize the average validation loss resulting from a stratified 10-times repeated 10-fold cross validation process. The maximum number of iterations was 300 for the dense sub-network and lowered to 100 for the convolutional sub-network, given its highly increased capacity. Each fold was run using early stopping based on the loss value on a withheld 10% portion of the training fold with a maximum of 2000 epochs (20 gradient updates). In hyperparameter tuning, models were optimized using SGD with a learning rate of $0.01$. The architecture with the highest Harrell’s concordance index [102] was selected.

Internal validation and external test. Internal model performance was assessed using 10 repetitions of stratified 10-fold cross-validation on the LVSPSCD cohort. Early stopping based on the c-index on a withheld 10% subset was implemented with a maximum training of 2000 epochs (20 gradient updates). The optimizer was Adam with learning rate $10^{-5}$ for the convolutional sub-network, $5 \times 10^{-4}$ for the dense sub-network, and $0.01$ for the ensemble. A final model was trained with all the available
LVSPSCD data and tested on the PRE-DETERMINE cohort. To estimate confidence intervals on the external cohort, the same cross-validation process was applied to the PRE-DETERMINE cohort, supplementing the training data in each fold with the LVSPSCD cohort. Approximate normal confidence intervals were constructed using the 100 folds.

**Gradient-based Interpretation of SSCAR** The trained network weights in SSCAR were interpreted for both dense and convolutional sub-network using the gradients of outputs with respect to intermediary neural network internal representations of data. For the convolutional sub-network, we adapted Grad-CAM [105] to work on regression problems and applied it to SSCAR by performing a weighted average of the last convolutional layer feature maps, where the weights were averages of gradients of the location parameter output with respect to each channel (see Appendix II for details). The result was then interpolated back to the original image dimensions and overlaid to obtain the gradient maps shown in Fig. 4-2a, bottom row. For the dense sub-network, the gradient of the location parameter output was taken with respect to each of the inputs and averaged over three groups: all patients, patients with SCDA, patients with no SCDA.

**Patient Population and Data Sets**

This study was a retrospective analysis based on a subset \( n = 269 \) of patients selected from the prospective clinical trials described below. Of note is that the entire model development in this manuscript was based on the internal cohort (see below), while the external cohort was used exclusively for testing (outcomes were solely used for computing relevant metrics once the model was fixed).

**LV Structural Predictors of SCD cohort (internal)** Patient data came from the Left Ventricular Structural Predictors of Sudden Cardiac Death Study (Clinical-
Figure 4-2. SSCAR Interpretation. The features learned by SSCAR are interpreted by performing a gradient-based sensitivity analysis of the location parameter to changes in the neural network input or features. The magnitude of the gradient measures the strength of the sensitivity of the predicted $T_{SCDA}$ to inputs or intermediary features. The sign of the gradient shows the direction of the effect. a. Shown is the convolutional sub-network feature interpretation for an example patient who did not experience SCDA (No SCDA, top) and for a patient who did (SCDA, bottom). For each patient, contrast-enhanced short-axis cardiac magnetic resonance images (shown here are a subset of 3 locations in the heart, base to apex, top to bottom, left column) used as inputs by SSCAR are overlaid with blood pool and myocardium segmentation (middle column, orange and green, respectively). A heat map of extracted features scaled by the value of the gradient shows contribution of the local pixel intensity to the predicted location parameter for the last convolutional layer (right column, blue and red heat maps). b. Dense covariate sub-network interpretation based on an average of all patients (solid bars), patients with SCDA (square hash bars), and no SCDA (diagonal hash bars). Top three highest and bottom three lowest average gradients of the neural network output (i.e. the predicted location parameter) with respect to the clinical covariate inputs are shown. Abbreviations: lvef_cmr, left ventricular ejection fraction from CMR; betablock, use of $\beta$-blocker medication; ekg_hr, heart rate from ECG; ekg_qrs_dur, QRS duration from ECG; lv_mass_ed, left ventricular mass in end diastole.
Trials.gov ID NCT01076660) sponsored by Johns Hopkins University. As previously
described[100, 106], patients satisfying clinical criteria for ICD therapy for SCDA
(LVEF ≤ 35%) were enrolled at 3 sites: Johns Hopkins Medical Institutions (Bal-
timore, MD), Christiana Care Health System (Newark, DE), and the University of
Maryland (Baltimore, MD). A total of 382 patients were enrolled between November
2003 and April 2015. Patients were excluded if they had contraindications to CMR,
New York Heart Association functional class IV, acute myocarditis, acute sarcoido-
sis, infiltrative disorders (e.g., amyloidosis), congenital heart disease, hypertrophic
cardiomyopathy, or renal insufficiency (creatinine clearance < 30 mL/minute after
July 2006 or < 60 mL/minute after February 2007). The protocol was approved by
the institutional review boards at each site, and all participants provided informed
consent. CMR images was performed within a median time of 3 days before ICD
implantation. The current study focused on the ischemic cardiomyopathy patient
subset with adequate late gadolinium enhanced (LGE)–CMR, totaling 156 patients.
As part of the clinical study, the participants had undergone single-chamber or dual-
chamber ICD, or cardiac resynchronization with an ICD (CRT-D) implantation based
on current guidelines. The programming of antitachycardia therapies was left to the
discretion of the operators.

**PRE-DETERMINE cohort (external).** Patients were selected from
PRE-DETERMINE (NCT01114269) and the DETERMINE Registry (NCT00487279),
sponsored by Brigham and Women’s Hospital. The populations were from multi-center
prospective cohort studies comprised of patients with coronary disease on angiography
or documented history of myocardial infarction (MI). The PRE-DETERMINE study
enrolled 5764 patients with documented MI and/or mild to moderate left ventricular
dysfunction (LVEF between 35–50%) who did not fulfil consensus guideline crite-
rria for ICD implantation on the basis of LVEF and New York Heart Association
class\textsuperscript{[107]}. Exclusion criteria included a history of cardiac arrest not associated with acute MI, current or planned ICD, or life expectancy < 6 months. The accompanying DETERMINE Registry included 192 participants screened for enrolment in PRE-DETERMINE who did not fulfil all criteria. The current study focused on 113 patients with adequate LGE–CMR who had similar covariate characteristics ("matched") to the internal cohort and would most closely satisfy the inclusion/exclusion criteria of the LVSPSCD cohort, although LVEFs were higher.

**LGE-CMR Acquisition**

The CMR images in the internal and external cohort were acquired using 1.5-T magnetic resonance imaging devices (Signa, GE Medical Systems, Waukesha, Wisconsin; Avanto, Siemens, Erlangen, Germany). All were 2-D parallel short-axis left ventricle stacks. The contrast agent used was 0.15 – 0.20 mmol/kg gadodiamide (Omniscan, GE Healthcare) or gadopentetate dimeglumine (Magnevist, Schering AG) and the scan was captured 10 – 30 minutes after injection. Due to the multi-center nature of the clinical studies considered here, there were variations in CMR acquisition protocols. The most commonly used sequence was inversion recovery fast gradient echo pulse, with an inversion recovery time typically starting at 250ms and adjusted iteratively to achieve maximum nulling of normal myocardium. Typical spatial resolutions ranged $1.5 - 2.4 \times 1.5 - 2.4 \times 6 - 8 \text{ mm}$, with $2 - 4\text{mm}$ gaps. More details regarding on CMR acquisition can be found in previous work\textsuperscript{[108, 109, 100, 106]}.

**Clinical Data and Primary Endpoint**

In both LVSPSCD and PRE-DETERMINE/DETERMINE cohorts, baseline data on demographics, clinical characteristics, medical history, medications, lifestyle habits, and cardiac test results were collected (see Table 4-II for a list of the common ones between the cohorts that were used in SSCAR). The primary endpoint for LVSPSCD was SCDA
defined as therapy from the ICD for rapid ventricular fibrillation or tachycardia, or a ventricular arrhythmia not corrected by the ICD. For PRE-DETERMINE, the primary end point was SCDA defined as resuscitated ventricular fibrillation, as determined from medical records pertaining to all deaths, cardiac arrests and ICD implantation, as well as interviews with family members. In both cohorts, phone interviews were conducted every 6 months to assess endpoints. Median follow-up time in this study was 6.5 years, with 41 patients (26%) reaching the endpoint, in LVSPSCD, and 8.6 years, with 22 patients (20%) reaching the endpoint, in PRE-DETERMINE.

Data Preparation

The inputs to our model were the unprocessed late gadolinium enhanced (LGE)-CMR scans and the clinical covariates listed in Table 4-II. The training targets were the event time and event type (SCDA or censoring). As a pre-processing step, the raw LGE-CMR scans were first segmented for LV myocardium using a method based on convolutional neural networks developed and described in previous work [12] (see Chapter 3). Briefly, this segmentation network consisted of 3 sub-networks: a U-net with residual connections trained to identify the entire region of interest, a U-net with residual connections trained to delineate the myocardium wall, and an autoencoder tasked with correcting anatomical inaccuracies that may have resulted in the segmentation. In this context, anatomically correctness was defined via a list of pass/fail rules (e.g., no holes in the myocardium, circularity threshold, no disconnected components, etc.). Once each patient’s LGE–CMR images were segmented via this method, all voxels outside the LV myocardium were zeroed out and the slices were sorted apex-to-base using DICOM header information and step-interpolated on a regular $64 \times 64 \times 12$ grid with voxel dimensions $2.5 \times 2.5 \times 10$ mm. These dimensions were chosen to make all patient volumes consistent with minimal interpolation from the original resolution, while allowing enough room to avoid truncating the LV. Finally,
### Table 4-II. Clinical Covariate Data.

Covariates are shown for both the internal and external cohorts. For continuous variables, the values are mean (± std. dev.) and for categorical, count (% of total). *P*-values are based on two-sample Welch’s t-test for continuous variables and Mann–Whitney U test for categorical variables. Abbreviations: SCDA, sudden cardiac death from arrhythmia; DM, diabetes mellitus; EF, ejection fraction; CMR, cardiac magnetic resonance; CM, cardiomyopathy; LVEF, left ventricular ejection fraction; LV, left ventricle; ED, end-diastolic; LBBB, left bundle branch block.

<table>
<thead>
<tr>
<th>Demographics</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age, y</td>
<td>61 (± 11)</td>
<td>62 (± 11)</td>
<td>0.443</td>
</tr>
<tr>
<td>Male sex</td>
<td>135 (87)</td>
<td>98 (87)</td>
<td>0.483</td>
</tr>
<tr>
<td>White</td>
<td>126 (81)</td>
<td>95 (84)</td>
<td>0.345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk Factors</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tobacco use</td>
<td>104 (67)</td>
<td>83 (73)</td>
<td>0.117</td>
</tr>
<tr>
<td>DM</td>
<td>51 (33)</td>
<td>44 (39)</td>
<td>0.146</td>
</tr>
<tr>
<td>Hypertension</td>
<td>105 (67)</td>
<td>79 (70)</td>
<td>0.326</td>
</tr>
<tr>
<td>Hyperlipidemia</td>
<td>121 (78)</td>
<td>106 (94)</td>
<td>p &lt; .001</td>
</tr>
<tr>
<td>EF non-CMR, %</td>
<td>25 (± 7)</td>
<td>39 (± 13)</td>
<td>p &lt; .001</td>
</tr>
<tr>
<td>Duration of CM, y</td>
<td>5 (± 6)</td>
<td>5 (± 7)</td>
<td>0.920</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMR Measurements</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVEF, %</td>
<td>28 (± 8)</td>
<td>36 (± 11)</td>
<td>p &lt; .001</td>
</tr>
<tr>
<td>LV mass (ED), g</td>
<td>146 (± 45)</td>
<td>127 (± 35)</td>
<td>p &lt; .001</td>
</tr>
<tr>
<td>Infarct Size, %</td>
<td>28 (± 14)</td>
<td>16 (± 10)</td>
<td>p &lt; .001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECG Measurements</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart rate</td>
<td>70 (± 12)</td>
<td>69 (± 14)</td>
<td>0.748</td>
</tr>
<tr>
<td>Presence of LBBB</td>
<td>24 (15)</td>
<td>5 (4)</td>
<td>0.002</td>
</tr>
<tr>
<td>History of atrial fibrillation</td>
<td>29 (19)</td>
<td>4 (4)</td>
<td>p &lt; .001</td>
</tr>
<tr>
<td>QRS duration, ms</td>
<td>116 (± 27)</td>
<td>104 (± 24)</td>
<td>p &lt; .001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Medication Use</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>β-Blocker</td>
<td>146 (94)</td>
<td>103 (91)</td>
<td>0.227</td>
</tr>
<tr>
<td>ACE inhibitor or ARB</td>
<td>141 (90)</td>
<td>94 (83)</td>
<td>0.040</td>
</tr>
<tr>
<td>Lipid-lowering</td>
<td>142 (91)</td>
<td>105 (93)</td>
<td>0.289</td>
</tr>
<tr>
<td>Diuretic</td>
<td>80 (51)</td>
<td>58 (51)</td>
<td>0.497</td>
</tr>
<tr>
<td>Antiarrhythmic Drug</td>
<td>13 (8)</td>
<td>2 (2)</td>
<td>0.010</td>
</tr>
<tr>
<td>Digoxin</td>
<td>20 (13)</td>
<td>2 (2)</td>
<td>p &lt; .001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Internal (n = 156)</th>
<th>External (n = 113)</th>
<th><em>P</em>-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCDA event</td>
<td>41 (26)</td>
<td>22 (19)</td>
<td>0.097</td>
</tr>
<tr>
<td>Time to event, y</td>
<td>6 (± 3)</td>
<td>7 (± 3)</td>
<td>0.004</td>
</tr>
</tbody>
</table>

The input to the neural network model consisted of a two-channel image. The first channel was a one-hot encoding of the myocardium and blood pool masks. The second channel had zeros outside of the myocardium and the original CMR intensities on the myocardium, linearly scaled by multiplication with half the inverse of the median blood pool intensity in each slice. To mitigate overfitting, train-time data augmentation was performed on the images, specifically 3-D in-plane rotations in increments of 90° to...
avoid artifacts and panning of the ventricle within the box. The clinical covariate data was de-meaned and scaled by the standard deviation.

**Statistical Analysis**

All values reported on the internal validation data set were averages over 100 data splits resulting from a 10-times repeated 10-fold stratified cross validation scheme. Values reported on the external test data set represented a single evaluation on the entire set. All confidence intervals were normal approximations resulting from the aforementioned 100 splits. In computing confidence intervals for the external test set, the same procedure was used on all available data, ensuring test folds came exclusively from the external data set. Error bars are standard errors with sample standard deviation estimated from the 100 splits. Correlation $P$-value was based on the exact distribution under the bivariate normal assumption. Covariate $P$-values are based on two-sample Welch’s t-test [110] for continuous variables and Mann–Whitney U test for categorical variables. Cox proportional hazards analysis was performed using the Python lifelines [111] package.

**Results**

**SSCAR Overview**

The arrhythmia prediction neural network in SSCAR is a deep learning framework that incorporates multiple custom neural networks (which fuse different data types) combined with statistical survival analysis, to predict patient-specific probabilities of SCDA at future time points. Fig. 4-1 presents an overview of SSCAR. On the left and right, CMR images and clinical covariates (yellow panel) are used as inputs to the two corresponding branches of the model. The goal of each of the branches is to predict the patient-specific survival curve. In the left branch, cardiac CMR images — visualizing the patients’ 3-D ventricle geometry and contrast-enhanced remodeled tissue — are
used as input by a custom-designed encoder-decoder convolutional neural sub-network (red panel, left). This sub-network is trained to discover and extract imaging features associated with SCDA risk directly from the CMR images. The encoder-decoder design of the sub-network ensures that resulting imaging features retain sufficient information to be able to reconstruct the original images (red panel, left, decoder path). In the right branch, 22 clinical covariates are provided to a dense sub-network (green panel, right), which discovers and extracts nonlinear relationships between the input variables. The outputs of the sub-networks are combined (ensembled) in a way that best fits the observed SCDA event training data (center path, dot-dashed) to estimate a pair of individualized parameters representing the most probable time to SCDA ($T_{SCDA}$) and a measure of uncertainty in the prediction. These two parameters determine, for each patient, a cause-specific survival curve (bottom, blue) within a given statistical model.

**SSCAR Overall Risk Prediction Performance**

SSCAR was developed and internally validated using data from 156 patients with ischemic cardiomyopathy (ICM) enrolled in the LV Structural Predictors of SCD (LVSPSCD) prospective observational study [100, 106]. To demonstrate the model’s performance out-of-sample, an external test was performed using a similar cohort of 113 patients with ICM and available CMR images (see Methods for details). Those were selected from the PRE-DETERMINE study [107], which comprised patients with less severe LV systolic dysfunction, but otherwise had similar inclusion/exclusion criteria to those in the LVSPSCD study. SSCAR performance was evaluated comprehensively using Harrell’s concordance-index (c-index) [102] — an estimate of the probability of assigning the correct risk order given 2 randomly selected patients (range is $[0, 1]$, and higher is better) — and the integrated Brier score ($\overline{Bs}$) [103] — an estimate of the squared difference between estimated survival probability and 0/1 outcome by a certain
time, averaged over all times (lower scores are better). Fig. 4-3a demonstrates that SSCAR has excellent concordance on the internal set (.82–.89) for events occurring before time \( \tau \), where \( \tau \) ranges from 2 to 10 years. Additionally, the \( \overline{B} \)s ranges from .04 to 0.12, indicating strong calibration. Despite the dissimilarities between cohorts, SSCAR performance carries over well to the external cohort, resulting in a c-index of 0.71 – 0.77 and \( \overline{B} \)s of .02 – 0.14. Remarkably, the model maintains its risk discrimination abilities at all times, as further evidenced by the results presented in Fig. 4-4. In Figs. 4-3b–c, all events up to 10 years are used to construct the receiver operator characteristic (ROC) and precision-recall (PR) curves. The area under the ROC curve is 0.87 (95% CI: 0.84 – 0.90) for the internal validation set and 0.72 (95% CI: 0.67 – 0.77) for the external test set. The area under the PR curve is 0.93 (95% CI: 0.91 – 0.95) for the internal set and 0.73 (95% CI: 0.68 – 0.78) for the external.

**Patient-Specific Survival Curves Predicted by SSCAR**

The SSCAR survival model presented here predicts entire cause-specific survival curves for each patient through two individualized parameters: the location \( \mu \) and scale \( \sigma \), characterizing the probability distribution of \( T_{SCDA} \) (see Methods for details). Using deep neural networks to directly learn these parameters from CMR images and from clinical covariates in a way that best models the survival data produces highly-individualized survival probability predictions. Fig. 4-5a illustrates individualized cause-specific survival curves (solid, blue) for a patient with \( T_{SCDA} \) around 6 years (left panel) and a patient censored (non-SCDA event) at around 7 years (right panel). In both cases, the survival curves estimated by SSCAR accurately predict the event probabilities: in the first case, the estimated survival probability crosses the 50% threshold close to the event time; in the censored case, SSCAR predicts > 80% probability of survival at the time of the (non-SCDA) event. For reference, two other survival curves are depicted: the Kaplan-Meier estimate (purple, dot-dashed) and
Figure 4-3. **SSCAR Overall Performance.**

**a.** Concordance index (top, blue) measuring model risk discrimination — higher is better — and integrated Brier score (bottom, red) showing calibration — lower is better — for various time points.

**b.** Receiver operator characteristic (ROC) curve at 10 years for the internal validation and external test cohorts, with the respective areas under the curve (AUROC).

**c.** Precision-recall (PR) curve at 10 years for the internal validation and external test cohorts, with the respective areas under the curve (AUPR). For all panels, shaded areas represent approximate 95% confidence intervals, solid and dashed lines indicate the internal and external cohorts, respectively, and random chance performance thresholds are shown using dotted lines. The chosen time of 10 years is larger than all sudden cardiac death from arrhythmia event times in the population.

The Breslow estimate based on a Cox proportional hazards model using the clinical covariates (green, dashed), demonstrating worse performance by underestimating the risk for the patient with SCDA and overestimating for the censored patient. Further details on SSCAR’s performance compared to the Cox proportional hazards model are presented in Table 4-III.
Figure 4-4. SSCAR Results at Various Time Points. Receiver operator curves (ROC) for years 2 – 9 for the internal and external cohorts, with the respective areas under the curve (AUROC). Predicted outcomes are based on the estimated survival probability at the respective time points as computed from the survival probability function. Shaded areas represent approximate 95% confidence intervals, solid and dashed lines indicate the internal and external cohorts, respectively, and random chance performance thresholds are shown using dotted lines.

The predicted location parameter estimates the most probable $T_{SCDA}$ and the predicted scale parameter provides a measure of confidence for the location. The inclusion of both a location and a scale parameter in the model offers the advantage of building in uncertainty directly into the $T_{SCDA}$ prediction. Importantly, this uncertainty is patient-specific and learned from data. Fig. 4-5b presents examples of predicted $T_{SCDA}$ probability distributions for two patients (P1 and P2) with different scale parameters, visualized as the widths of the distributions. Shown are the actual (dotted) and predicted (solid) $T_{SCDA}$, as well as the probability distributions (shaded). For P1, the prediction error is small (solid vs. dashed vertical lines) and the model is certain, as seen by the narrower probability distribution of P1’s $T_{SCDA}$, or, equivalently, a smaller predicted scale parameter. In the case of P2, the prediction error is larger and the model predicts a wider distribution, or, equivalently, a larger scale parameter, indicating higher uncertainty. Remarkably, using the entire cohort to
quantify this direct relationship between prediction error — calculated as the relative mean absolute difference of actual and predicted times — and scale parameter reveals significant positive correlation (Pearson’s $r = 0.42$, $p < 0.001$), demonstrating that SSCAR recognizes which predictions of $T_{SCDA}$ will turn out inaccurate and “lowers the confidence” in them through a larger scale parameter.

<table>
<thead>
<tr>
<th>Model Comparison</th>
<th>c-index</th>
<th>Brier Score</th>
<th>BA</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Cox PH</td>
<td>0.58 (0.55 – 0.61)</td>
<td>0.30 (0.29 – 0.32)</td>
<td>0.45 (0.40 – 0.49)</td>
<td>0.69 (0.65 – 0.72)</td>
</tr>
<tr>
<td>SSCAR (cov. only, Cox survival)</td>
<td>0.73 (0.71 – 0.76)</td>
<td>0.14 (0.13 – 0.15)</td>
<td>0.65 (0.60 – 0.69)</td>
<td>0.78 (0.75 – 0.81)</td>
</tr>
<tr>
<td>SSCAR (cov. only)</td>
<td>0.71 (0.68 – 0.73)</td>
<td>0.16 (0.16 – 0.17)</td>
<td>0.57 (0.53 – 0.62)</td>
<td>0.79 (0.75 – 0.82)</td>
</tr>
<tr>
<td>SSCAR (CMR only)</td>
<td>0.70 (0.67 – 0.72)</td>
<td>0.17 (0.17 – 0.18)</td>
<td>0.60 (0.55 – 0.66)</td>
<td>0.81 (0.77 – 0.85)</td>
</tr>
<tr>
<td>SSCAR</td>
<td>0.83 (0.81 – 0.85)</td>
<td>0.12 (0.12 – 0.13)</td>
<td>0.77 (0.73 – 0.81)</td>
<td>0.87 (0.84 – 0.89)</td>
</tr>
</tbody>
</table>

Table 4-III. Model Comparison. Concordance index (c-index), integrated Brier score (Brier Score), balanced accuracy (BA) and F-score (see Methods for description) are shown for a standard Cox proportional hazards model fit on the clinical covariates (Linear Cox PH), the dense sub-network of SSCAR using clinical covariates with a Cox survival model (cov. only, Cox survival), the dense sub-network of SSCAR using clinical covariates and the log-logistic survival model (cov. only), the convolutional sub-network using cardiac magnetic resonance images only (CMR only) and the full arrhythmia prediction neural network model (SSCAR). All performance measures are calculated at $\tau = 10$ years. All model comparison values are based on averages over 100 cross-validation splits of the internal validation data set. In parentheses, approximate 95% confidence intervals are shown (see Methods for details).

Image-based Risk Prediction

The convolutional sub-network (see Fig. 4-6 for architecture details) in SSCAR integrates neural network DL on images within an overall statistical survival model. This branch of SSCAR uses LGE-CMRs — a modality uniquely suited for visualizing ventricle geometry and portions of the myocardium with contrast-enhanced remodelling — to learn image features most useful in predicting a patient’s survival $T_{SCDA}$. CMR raw pixel values are directly provided to the network, after which the left ventricle is automatically segmented, eliminating the need for arbitrary thresholds aiming to delineate areas of enhancement. Table 4-IV shows detailed risk prediction performance for the convolutional sub-network branch (CMR only) model. Using only images as inputs, SSCAR achieves 0.70 (95% CI: 0.67–0.72) c-index and 0.17 (95% CI: 0.167–
Figure 4-5. Individual Patient Survival Probability. a. Survival probability curves are shown for an example patient in the external test set who experienced sudden cardiac death from arrhythmia (SCDA, left display), and for one who did not (No SCDA, right display). Survival probability curves are plotted over time for SSCAR (solid, blue), Cox proportional hazards (Cox PH) model (dashed, green) on the clinical covariates, Kaplan-Meier estimator (dot-dashed, purple), together with the indicator ground truth (dotted, black). For the patient with SCDA, SSCAR crosses the 50% survival probability threshold significantly closer to the SCDA time, as compared to the alternative curves, highlighting the model’s high calibration. For the censored patient (no SCDA), SSCAR estimates higher survival probability at the time of non-SCDA event compared to the other models.

b. Examples of SSCAR’s predicted probability distributions for the time to SCDA (shaded areas, pdf($T_{SCDA}$)) for two patients in the external test set who experienced SCDA (P1, blue and P2, orange). The predicted times to event (Predicted $T_{SCDA}$) are depicted as solid vertical lines (peaks of distributions); actual times (Actual $T_{SCDA}$) are depicted by dotted vertical lines. Note that SSCAR has a larger prediction error in P2 compared to P1, seen on the graph as the distance between the respective solid and dotted lines. However, SSCAR “recognizes” the inaccurate $T_{SCDA}$ prediction and compensates for that by also predicting a more spread out distribution (larger scale parameter) for P2. This direct relationship between the prediction error and predicted scale parameter holds more generally for the entire dataset, suggesting SSCAR learns to quantify the degree of inaccuracy in the $T_{SCDA}$ prediction.

0.178) BSs for event data truncated at 10 years on the internal validation set. On the external testing set, the CMR only model achieves 0.63 (95% CI: 0.59–0.66) c-index and 0.19 (95% CI: 0.186–0.200) BSs. It is noteworthy that, although the dense sub-network uses 22 clinical covariates and already includes manually engineered features...
from the CMRs (e.g., infarct fraction), the convolutional sub-network using only CMR as inputs achieves similar performance to the dense one, as shown in Table 4-III. Furthermore, assembling the two sub-networks together leads to a significant increase in overall performance compared to using just the covariate-based one, demonstrating that the convolutional sub-network identifies completely different CMR-based features than the manually engineered ones.

Imaging features learned by the convolutional network can be interpreted using a gradient-based sensitivity analysis as illustrated in Fig. 4-2a. The gradient here quantifies the sensitivity of the predicted $T_{SCDA}$ to the convolutional features learned by the neural network. Each resulting feature is scaled by the appropriate gradient to form the gradient map (see Methods for details). This map overlaid on the myocardium (right column, blue and red heatmap) shows the degree of contribution of the local pixel intensity to the most probable $T_{SCDA}$ (i.e., to the location parameter) for a patient without an SCDA event (top) and one with SCDA (bottom). Myocardial regions found to be characterized with large positive gradient (dark blue) are interpreted as having high importance in increasing $T_{SCDA}$ and, conversely, regions with large magnitude negative gradient (dark red) represent areas that are responsible for decreasing the predicted $T_{SCDA}$. The areas of contrast-enhanced myocardium (middle column in brighter green) do not fully overlap with the gradient map, which suggests that while features learned by the convolutional neural network may co-localize with enhanced tissue, the algorithm does not act as a mere enhancement locator. For example, the patient who did not experience SCDA has contrast-enhanced tissue, but the effect of these regions is to increase the predicted $T_{SCDA}$, suggesting a nuanced relationship between presence of enhancement and propensity of SCDA.
Figure 4-6. **SSCAR Convolutional Sub-network.** Top-left panel (yellow) shows patient data used in by the convolutional sub-network. The convolutional sub-network uses as input contrast-enhanced (LGE) cardiac magnetic resonance (CMR) images with the left ventricle automatically segmented (top inset). Labels associated with each patient (SCDA Events, bottom inset, dot-dashed contour) — consisting of the observed time to event, and an indicator whether the event was sudden cardiac death from arrhythmia (SCDA) or non-SCDA — are used as targets during training only. LGE-CMR data is taken as input by a 3-D convolutional neural network constructed using an encoder-decoder architecture (red panel, left). The encoder branch consists of a sequence of 3-D convolutions, downsampling (maxpool), and rectified linear unit (ReLU) activation functions. The encoded version of the image is transformed using a dense layer with a custom survival activation function into two parameters (location, $\mu$ and scale, $\sigma$, bottom panel, blue) for each patient, which fully characterize the probability distribution function of the patient-specific time to SCDA. The decoder branch uses transpose convolutions to reconstruct the original images, thereby ensuring that the encoded version of the CMR data is meaningful and able to reproduce the original images. During training (dot-dashed arrows), the neural network weights are optimized via a likelihood-based loss and a reconstruction loss.

**Nonlinear Dense Neural Network for Covariate Data**

SSCAR incorporates patient clinical covariate data through the use of a dense multi-layer neural network (Fig. 4-1, green panel). This sub-network discovers and extracts
potential nonlinear relationships between the covariates and integrates them within SSCAR’s overall survival predictions. In Table 4-III, we demonstrate the utility of the sub-network by comparing its performance with a (linear) Cox proportional hazards model. To avoid misattributing performance differences to the underlying statistical models, we replace the neural network loss function that used in a Cox proportional hazards model. Using clinical covariate data only, SSCAR with a Cox survival model (cov. only, Cox survival) outperforms the standard Cox proportional hazards model (Linear Cox PH) in terms of c-index (0.73 vs. 0.58) and Bs (0.14 vs 0.30), balanced accuracy (0.65 vs. 0.45), and F-score (0.78 vs 0.69). In Fig. 4-2b, we show that the neural-network model maintains interpretability by performing a sensitivity analysis of the predicted $T_{SCDA}$ with respect to changes in the covariates. As above, high positive gradients (blue) denote covariates for which small increases in their values lead to large increases in $T_{SCDA}$; whereas small negative gradients (red) represent covariates for which small increases lead to large decreases in $T_{SCDA}$. The top 3 positive gradient covariates are left ventricular ejection fraction computed from CMR, β-blocker medication, and heart rate computed from ECG. The bottom 3 negative gradient covariates are left ventricular mass at end-diastole, use of diuretic medication, and QRS duration computed from ECG.
**Discussion**

In this study we present a radically different and novel approach to SCDA risk assessment, the SSCAR DL framework, which uses a neural network survival model to predict patient-specific survival curves in ischemic cardiomyopathy. SSCAR consists of two neural networks, a 3-D convolutional network learning on raw unsegmented LGE-CMR images that visualize post-infarct scar distribution, and a fully-connected network operating on clinical covariates. SSCAR’s predicted patient-specific survival curves offer accurate SCDA probabilities at all times up to 10 years. SSCAR is not only a highly flexible model, able to capture complex imaging and non-imaging feature inter-dependencies, but is also robust owing to the statistical framework governing the way these features are combined to fit the survival data. Our framework predicts entire probability distributions for the $T_{SCDA}$, allowing for uncertainties in predictions to be themselves patient-specific and learned from data, thereby equipping the model with a self-correction mechanism. This approach remedies a well-known significant limitation of neural networks, the high confidence in erroneous predictions. SSCAR’s integration of deep neural network learning within a survival analysis and the resulting detailed outputs could represent a paradigm shift in the approach to SCDA risk assessment.

DL on LGE-CMR presented in this work is first-of-its-kind. Despite many heralding DL as the arrival of the artificial intelligence age in personalized healthcare\cite{5, 6, 7, 8, 9}, no significant progress has so far been made using DL on contrast-enhanced cardiac images to assess arrhythmia risk. Although there have been non-DL efforts to incorporate clinical imaging-derived features in SCDA risk stratification\cite{112, 113, 114}, these severely underutilize the data, suffering from two main limitations: features often rely on time-consuming, manual processing steps, typically involving arbitrarily chosen image intensity thresholds; or features are either too coarse to capture the intricacies of the scar distribution, or highly mathematical, undermining their physiological
underpinning. On the other hand, the DL efforts related to arrhythmia have focused primarily on its cardiologist-level detection in ECG signals [115, 116, 117, 118, 119]. In the current work, we present for the first time a DL approach which takes as input directly raw, unsegmented LGE-CMR images and automatically identifies features which best model and predict the $T_{SCDA}$.

SSCAR is the first SCDA risk prediction model to combine raw imaging with other data types in the same framework. Our technology operates on LGE-CMR images and clinical covariates within a unified feature learning process, allowing for the different data types to synergistically inform the overall survival model. Among the clinical covariates used in SSCAR are standard manually derived imaging features, which prevents the convolutional neural network from merely re-discovering these known features, and instead encourages it to learn new features. SSCAR achieves performance that is beyond the state-of-the-art in both relative terms — SCDA risk ordering among patients — as well as absolute — accurately calibrated probabilities of SCDA. Our robust testing scheme overcomes significant limitations of previous work on SCDA risk prediction [99, 112, 107, 120, 113]. First, we demonstrate high generalizibility by computing internal cross-validation performance numbers resulting from 100 train/test splits of the data and, importantly, on an entirely separate external cohort from a different study, showing modest performance degradation. Second, our approach prevents the model from being over-tuned to a certain time horizon by computing performance metrics at multiple time points up to 10 years.

Since SSCAR is a combination of neural networks, each working on different data types (images and clinical covariates), we were able to perform a comprehensive bottom-up analysis of overall performance. We demonstrated that the added complexity of our DL approach — potentially at some expense to interpretability — is justified by the significantly elevated performance numbers. Indeed, we developed and evaluated a regularized Cox proportional-hazards using the available clinical covariates to serve
as a baseline for the rest of the analysis. We showed that the neural network-driven feature extraction of SSCAR on the same covariates performs significantly better in the same proportional-hazards setting, highlighting the importance of nonlinear relationships in the covariates. Furthermore, we showed that even when using only LGE-CMR images to predict arrhythmia risk, the convolutional neural network in SSCAR 1) outperforms the Cox proportional-hazards model constructed using clinical covariates which include standard imaging and non-imaging features, and 2) performs on par with the covariate-only network in SSCAR using the same clinical variables, suggesting that the image-only neural network in SSCAR is able to identify highly predictive imaging features in the LGE-CMR images. Finally, we demonstrate that the novel imaging features found by SSCAR’s convolutional network cannot be explained away even when considering nonlinear relationships between standard covariates, as evidenced by the ensembled SSCAR model superior performance over SSCAR using either data type.

Importantly, a level of interpretability is embedded in the overall design of the custom neural network used in SSCAR. Interpretability of AI algorithms is paramount to their broad adoption and concerns surrounding it are particularly prevalent in healthcare. In our approach, we take multiple steps to ensure the relevance and interpretability of resulting features. Our sensitivity analysis of the outputs to the extracted features offers a lens into the neural network, rendering some transparency to the algorithm “black-box”. In addition, CMR images taken as input by the convolutional neural network are automatically segmented to include myocardium-only raw intensity values and the network is designed as an encoder-decoder to ensure minimal loss of information during the feature extraction process.

SSCAR achieves excellent performance despite working on a relatively small data set. A concern with DL on smaller data sets is overfitting, which manifests itself as high performance during training (good fit), but poor performance when applied to a
new test set. Indeed, the results in this paper show some differences between metrics on the internal validation and external test cohorts. However, we emphasize that although the two cohorts’ covariates were “matched” where possible (see Methods), they represent two different distributions (e.g., low vs. moderately reduced LVEF). Furthermore, several measures were taken to mitigate overfitting: in addition to standard techniques — dropout, kernel and bias regularizers — we designed the convolutional sub-network as an autoencoder which uses the distilled features used in risk prediction to also re-construct the original image as an additional regularization technique. Finally, all numbers cited on the internal validation set are averages of the test performance of hundreds of train/test data splits, adding a layer of statistical rigour.

The survival model used in SSCAR is also novel and incorporates recent thinking in estimating SCDA risk probabilities in the presence of competing risks. A potential shortcoming of models which do not directly model competing risks is that predicted probabilities for the event of interest assume a reality where no other type of death could occur, thereby undermining interpretability. In SSCAR, we directly model the cause-specific hazard rate and use the implied survival function to make predictions. A limitation here is that we could not compute the cause-specific cumulative incidence function, as it requires additional all-cause mortality data. However, should such data become available, our framework makes such an extension straightforward.

An additional limitation in this work is that the list of covariates is not comprehensive. Few standard clinical covariates were dropped when “matching” the internal and external cohort. However, since no standard imaging covariates were excluded, we do not expect any of the omitted variables to affect conclusions drawn regarding the performance of the sub-components of SSCAR relative to the baseline Cox model. Including additional covariates identified in past work as predictors of SCDA, but not part of standard clinical practice, was beyond the scope of our work. However,
these could in principle erode the performance of the image-based feature extraction in SSCAR in favor of the covariate-only part. Nevertheless, we would expect that, in general, including more variables with proper regularization can only improve the overall results in SSCAR, even if a re-balance of its components’ performance contribution occurs.

SSCAR fuses cutting-edge DL technology with modern survival analysis techniques. It represents innovation in CMR imaging feature extraction and learning of nonlinear relationships between standard clinical covariates. The technology aims to transform clinical decision-making regarding arrhythmia risk and patient prognosis by encouraging practitioners to eschew the view of predicted risk as a single number outputted by a “black-box” algorithm, but rather be guided by the estimated time-to-outcome in the context of patient-specific time prediction uncertainty, which is itself built in SSCAR’s learning process. Through its highly accurate predictions and high levels of generalizability and interpretability, SSCAR represents an essential step towards bringing patient trajectory prognostication into the age of artificial intelligence.
Conclusion and general discussion

In this work, we showed that state-of-the-art deep learning techniques can be integrated with statistical learning methods in a way that fully utilizes the unparalleled feature extraction of machine learning to build statistically rigorous models. We demonstrated our novel approaches by tackling two important problems in cardiac image segmentation and disease prediction using real-world clinical data. In Chapter 3, our deep learning technology for segmenting contrast-enhanced cardiac images provides an elegant solution to not only delineating regions of interest to practitioners (left ventricle myocardium, blood pool, area of contrast enhancement), but also showed how these can be used to seamlessly compute clinical measures (e.g., scar burden, myocardial volume), which are critical in the assessment of heart function. Obtaining anatomically correct segmentations is key to accurately computing clinical measures. Our convolutional encoder-decoder network was designed specifically to assure anatomical correctness by relying on neural network-driven encoding and statistical learning of what it means for a myocardial mask to be "anatomically correct". Our synergistic approach allowed us to use Gaussian mixture modeling on the latent space to further “densify” the autoencoder latent space, thereby improving overall prediction of the post-processing neural network.

In Chapter 4, we developed a new survival model which, unlike standard approaches, does not require manual selection of covariates. Instead it uses a deep learning architecture to extract features from both numerical clinical covariate data, as well as from raw, unsegmented, contrast-enhanced cardiac magnetic images. Additionally, it
uses modern neural network optimization techniques to maximize the profile likelihood of the survival data during the same process as the feature extraction one, thus shaping the latter to best fit the survival data. The framework described also capitalizes on recent techniques involving survival analysis in the presence of competing risks. We incorporate competing risks in our model and show how all-cause mortality data could be utilized to compute cumulative incidence functions used in disease prognosis.

While specific discussion on the applications described above are provided in the respective chapters, we now discuss the common thread between the two approaches. In both models, novel deep learning techniques are combined with more established standard statistical ones to create models aimed provide “the best of both worlds”. The capabilities of deep neural networks to identify features in images cannot be understated. There is no doubt that deep neural networks have quickly come to dominate many computer vision competitions, as well as helped shape the direction of tasks like natural language processing, speech recognition and, more recently, medical image analysis. It seems that while the mathematical theory backing modern complex neural network models has lagged, tremendous effort has gone in solving practical, day-to-day problems. As presented in Chapter 2, there exist theoretical results for simple neural networks, but typically only for simple models or for the — often unrealistic — “large n” case. Having a solid mathematical foundation when the size of the data are large and performing well in tasks where data are abundant (e.g., ImageNet classification problems), it is tempting to jump to the potentially erroneous conclusion that neural networks are not suited for small data sets. Indeed, due to their sometimes massive capacity, neural networks can very quickly overfit the training data, providing very poor generalization. However, writing off neural networks from fields like medicine — where data are oftentimes very scarce, particularly when they describe long-term patient follow-up — because they have proven effective in big data settings would not only be a logical fallacy, but would also do a disservice to the field.
By combining deep learning approaches and “taming” them within standard statistical models, we have shown how neural networks can successfully be used in real-world applications with scarce data.

An additional concern with the application of “off-the-shelf” deep neural network models, particularly in the medical field, is that they are often viewed as “black boxes”, lacking interpretability. While the driver of an autonomous car may not care about what the car is doing specifically in order to avoid pedestrians, a clinician might be weary of intervening in the case of a patient at risk because a machine learning algorithm assigned a high score. Combining neural network feature extraction with traditional models which offer better explainability could provide the necessary compromise to bring deep learning to medicine more generally. In Chapter 4, we show that the parametric survival model can be used to understand how the various features extracted using neural networks impact not only the predicted time-to-event, but also quantify uncertainty around these predictions. Having artificial intelligence that is compatible with our own and allows us to understand it in terms of our own knowledge and intuition seems sufficient for the broad adoption of machine learning, but whether it is necessary remains unclear. If we were certain that a machine learning algorithm makes 100% accurate predictions, one could reasonably ask if we actually cared about how it did it for reasons other than curiosity. Of course, in a world of imperfect artificial intelligence applications, perhaps grounding them in statistical models which help us gain intuition remains our best bet.

The idea of integrating deep learning within other fields, as opposed to merely using out-of-the-box pre-trained networks, is not new and has proven successful in many areas. For example, Raissi et al. [121] showed how deep learning can help with solving forward and inverse problems in Physics. Similarly, deep neural networks were used by Sirignano and Spiliopoulos [122] to solve high dimensional partial differential equations. Dai et al. [123] successfully used neural networks to help with NP-hard
combinatorial optimization problems. More recently, Katzman et al. [124] generalized the Cox proportional hazards model by forgoing the standard linearity assumption in the covariates and instead modeling this as a neural network.

Despite the vast body of work involving deep learning and its integration in other fields, there are significant limitations to consider. As previously mentioned, high capacity neural networks have a tendency to quickly overfit the training data. Even when using validation data for early stopping conditions, they have a propensity to learn the validation set. Additionally, it remains unclear how conclusions drawn from the mathematical theory developed on simple networks often involving limiting results extrapolate to real-world applications where data is much noisier and scarce. Without a clear understanding of such limitations, probabilities of error will be hard to estimate, which will jeopardize their widespread use. Much work has been done in shining a light in the deep learning “black box” [125], which would help with deep learning explainability. However, the consensus on which method is appropriate is lacking to the same degree as the consensus of what “explainable” means. However, these limitations should not act as a deterrent to more research in the field. While short-term exuberance should be tempered, the long-term prospects of artificial intelligence as an integral part of applied mathematics remain outstanding.

We have demonstrated how current limitations of deep learning can be overcome, by fusing artificial neural networks with established statistical learning models, thereby achieving cutting-edge feature extraction without compromising statistical rigor and interpretability. These approaches were used to solve two important real-world problems in Computational Cardiology, semantic segmentations of contrast-enhanced cardiac magnetic resonance imaging and prediction of survival in patients at risk of arrhythmic death. Our approaches achieved great test performance, despite working on relatively limited data sets. Such hybrid models could pave the way for elevating traditional statistical techniques and discovering new machine learning approaches to
data-intensive tasks across many disciplines.
Appendix I

Performance Metrics

A. Hausdorff Distance

Hausdorff distance [86] is used in the segmentation problem in Chapter 3. Let X and Y be two non-empty subsets of a metric space \((M, d)\). We define the Hausdorff distance \(d_H(X, Y)\)

\[
d_H(X, Y) = \max \left\{ \sup_{x \in X} d(x, Y), \sup_{y \in Y} d(X, y) \right\},
\]

where \(d(a, B) = \inf_{b \in B} d(a, b)\) for an element \(a \in X\) and a subset \(B \subseteq X\).

B. Concordance

The original version of the concordance index proposed by Harrell [102] is given by

\[
C_H^{\text{index}} = \frac{\sum_{i,j} \delta_i \mathbb{1}\{\eta_i > \eta_j\} \mathbb{1}\{x_i < x_j\}}{\sum_{i,j} \delta_i \mathbb{1}\{x_i < x_j\}},
\]

where \(\eta_i\) and \(\eta_j\) are the predicted risk scores associated with patients \(i\) and \(j\), \(x_i\) and \(x_j\) are the observed times (that is, realization of \(\min\{T, C\}\)), and \(\delta_i\) is the censoring status for patient \(i\). This estimator converges to the probability

\[
P\{\eta_1 > \eta_2 | T_1 < T_2, T_1 \leq \min\{C_1, C_2\}\},
\]

that is, the conditional probability that for 2 randomly selected individuals, the risk score of the first is greater than the second conditioned on the time to event for the
first being less than that of the second and the time to the first event is less than both censoring times. This estimator depends on the censoring distribution and, since the support of the censor time distribution tends to be shorter, estimation of the tail of the survival probability for $T$ suffers. To alleviate this, Pencina and D’Agostino[126] proposed the following estimator of concordance

$$
C_{PDA}^{\text{index},\tau} = \frac{\sum_{i,j} \delta_i \mathbb{1}\{\eta_i > \eta_j\} \mathbb{1}\{x_i < x_j, x_i < \tau\}}{\sum_{i,j} \delta_i \mathbb{1}\{x_i < x_j, x_i < \tau\}},
$$

which is an estimator for

$$
P\{\eta_1 > \eta_2|T_1 < T_2, T_1 \leq \min\{C_1, C_2\}, T_1 < \tau\}.
$$

This also depends on the censoring distribution. Instead, Uno et al.[104] propose the estimator

$$
C_{U}^{\text{index},\tau} = \frac{\sum_{i,j} \delta_i \left[\hat{G}(x_i)\right]^{-2} \mathbb{1}\{\eta_i > \eta_j\} \mathbb{1}\{t_i < t_j, t_i < \tau\}}{\sum_{i,j} \delta_i \left[\hat{G}(x_i)\right]^{-2} \mathbb{1}\{t_i < t_j, t_i < \tau\}},
$$

where $\hat{G}(\cdot)$ is the Kaplan-Meier estimator for the probability of censoring, that is $G(t) = P\{C > t\}$. They let

$$
C_{\tau} = P\{\eta_1 > \eta_2|T_1 < T_2\}
$$

and show that

$$
W = \frac{C_{U}^{\text{index},\tau} - C_{\tau}}{n}
$$

converges asymptotically to a normal random variable with mean 0. Throughout the analysis in Chapter 4, “concordance index” refers to the form given by $C_{U}^{\text{index},\tau}$.

### C. Integrated Brier Score

To evaluate performance at all times in predicting the survival curves in Chapter 4, we use the “integrated Brier score”. First, we define the Brier score at time $t$.

$$
B(t) = \frac{1}{n} \sum_{i=1}^{n} \left\{ \begin{array}{ll}
\frac{[0-S(t)]^2}{\hat{G}(t_i)} & \text{if } t_i < t, \delta_i = 1 \\
\frac{[1-S(t)]^2}{\hat{G}(t_i)} & \text{if } t_i > t \\
0 & \text{otherwise}
\end{array} \right.,
$$

110
where $\hat{S}(\cdot)$ is the predicted survival function and $\hat{G}(\cdot)$ is the Kaplan-Meier estimator of the censoring time distribution. Then, the integrated score is

$$\overline{Bs}(\tau) = \frac{1}{\tau} \int_0^\tau B(t)dt.$$
Appendix II

Noteworthy Code Excerpts

In this appendix, we present several noteworthy excerpts of Python code used in the development of the model.

A. Code from Chapter 3

a. Morphological Checks

```python
import cv2
import numpy as np

def _get_greatest_defect(contour, ch_idx):
    defects = cv2.convexityDefects(contour, ch_idx)
    if defects is not None:
        return np.max(defects[:, :, 3]) / 256.0
    else:
        return 0

def _get_equi_diameter(ch_contour):
    area = cv2.contourArea(ch_contour)
    equi_diameter = np.sqrt(4 * area / np.pi)
    return equi_diameter

def _convexity_defect(contour, ch_idx):
    greatest_defect = _get_greatest_defect(contour, ch_idx)
    if greatest_defect:
        equi_diameter = _get_equi_diameter(contour[np.squeeze(ch_idx)])
    else:
        equi_diameter = 1
```
return greatest_defect / equi_diameter

def is_c_shaped(mask):
    """
    Function to check if mask is C-shaped. Gets convexity defects and compares
greatest distance of defect to
diameter of convex hull.
:param mask: uint8 myocardial mask
:return: bool
    """
mask = np.asarray(mask, np.uint8)
contour, ch_idx = _get_convex_hull_contour(mask)
return _convexity_defect(contour, ch_idx) > .5

def has_thickness(mask, verbose):
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))
eroded = cv2.erode(mask, kernel)
is_whole = has_one(eroded, 0)

if verbose and not is_whole:
    print('Thickness constraint failed')
return is_whole

def is_circular(contour, verbose):
    """
    Returns whether object passes circularity threshold
:param contour: Contour of convex hull
:param verbose:
:return:
    """
area = cv2.contourArea(contour)
perim = cv2.arcLength(contour, True)
quiry = 4 * np.pi * area / perim ** 2

is_circ = quiry > OARITY_THRESHOLD

if verbose and not is_circ:
    print('Circularity constraint failed with %.2f' % quiry)
return is_circ

def _is_convex(contour, ch_idx, verbose):
    conv_defect = _convexity_defect(contour, ch_idx)
is_conv = conv_defect < DEFECT_THRESHOLD

if verbose and not is_conv:
    print('Convexity constraint failed with defect %.2f' % conv_defect)
return is_conv
def _no_components(mask, connectivity):
    no_comp = cv2.connectedComponents(mask, connectivity=connectivity)[0] - 1  # 1 component is background
    return no_comp

def has_one(mask, verbose):
    no_components = _no_components(mask, 8)
    h1 = no_components == 1

    if verbose and not h1:
        print('One myocardium constraint failed with %d' % no_components)
    return h1

def _has_no_holes(mask, is_c, verbose):
    reverse_mask = cv2.bitwise_not(mask)

    hnh = _no_components(reverse_mask, 4) <= 1 + (not is_c)  # Outside and inside of myo

    if verbose and not hnh:
        print('No holes constraint failed')
    return hnh

def _get_convex_hull_contour(mask):
    """
    Returns the contour of the mask together with points on its convex hull
    :param mask:
    :return:
    """
    closed_mask = mask.copy()
    _, contours, _ = cv2.findContours(closed_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    convex_hull = cv2.convexHull(contours[0], returnPoints=False)

    return contours[0], convex_hull

def minimum_area(mask, verbose):
    area_pct = np.sum(mask > 0) / mask.size
    if area_pct < MIN_AREA_PCT:
        if verbose:
            print('Minimum area constraint failed: %.2f' % area_pct)
        return False
    else:
        return True
def is_anatomically_correct(mask, verbose=0):
    mask_cp = mask.copy()
    mask_cp = np.squeeze(np.asarray(np.where(mask_cp > .5, 255, 0), np.uint8))

    # Certain checks aren't even worth doing if more than one myocardium
    if not minimum_area(mask_cp, verbose):
        return False
    if not has_one(mask_cp, verbose):  # number of components(3)
        return False
    else:
        if not has_thickness(mask_cp, verbose):  # requires has_one (therefore components(3))
            return False
        else:
            contour, ch_idx = _get_convex_hull_contour(mask_cp)
            ch_contour = contour[np.squeeze(ch_idx)]
            if not is_circular(ch_contour, verbose):  # convex hull contour
                return False
            else:
                is_c = _convexity_defect(contour, ch_idx) > .5
                if not _has_no_holes(mask_cp, is_c, verbose):
                    return False
                elif not is_c and not _is_convex(contour, ch_idx, verbose):  # c-shaped -> convex hull (2 ways)
                    return False

    return True

b. Latent Samples Generation

def generate_latent_samples(x_data, gaussian_components, model_save_path, n_samples=10000):
    """
    Take existing latent space, fit GMM and sample from it given they are anatomically correct
    :param x_data: data to fit as n x p
    :param n_samples: how many samples to draw
    :param gaussian_components: how many Gaussians to use
    :return:
    """

    gmm = GaussianMixture(n_components=gaussian_components, random_state=config.seed, n_init=10)
    gmm.fit(x_data)

    correct_samples = []
    while len(correct_samples) < n_samples:
        latent_sample = gmm.sample(n_samples=n_samples - len(correct_samples))[0]
        gmm.random_state = config.seed + len(correct_samples)
        decoded_sample = decode_anatae(latent_sample, model_save_path=model_save_path)
for i in range(latent_sample.shape[0]):
    if is_anatomically_correct(decoded_sample[i]):
        correct_samples.append(latent_sample[i])

correct_samples = np.asarray(correct_samples)

return correct_samples

B. Code from Chapter 4

a. Bespoke survival activation

import keras.backend as kbe
def loglogistic_activation(mu_logsig):
    
    Activation which ensures mu is between -3 and 3 and sigma is such that
    prediction is not more precise than 1 / n of a year.
    :param mu_logsig:
    :return:
    
    n = 12 # 1 / n is the fraction of the year in which at least p quantile of the
distribution lies
    p = .95 # quantile
    mu = kbe.clip(mu_logsig[:, 0], -3, 3)
    sig = kbe.exp(mu_logsig[:, 1])
    thrs = kbe.log((1 / (2 * n)) * (kbe.exp(-mu) + kbe.sqrt((2 * n) ** 2 + kbe.exp
                  (-2 * mu)))) / \n    kbe.log(kbe.cast_to_floatx(p / (1 - p)))

    logsig = kbe.log(thrs + kbe.relu(sig - thrs))

    mu = kbe.reshape(mu, (kbe.shape(mu)[0], 1))
    logsig = kbe.reshape(logsig, (kbe.shape(logsig)[0], 1))

    new = kbe.concatenate((mu, logsig), axis=1)
    return new

b. Survival loss function

import keras.backend as kbe
def logistic_nll_loss(mu, log_sigma, x, d):
    
    Loss function based on the negative log likelihood of a logistic distribution
    with
    right censored data.
    :param mu: tensor of the location latent parameter
:param log_sigma: tensor of the scale latent parameter
:param x: tensor of \(\min(\log(t), \log(c))\), \(t\) is event time and \(c\) is censor time
:param d: tensor of indicator \(t < c\)
:return:

```python
x_scaled = (x - mu) / kbe.exp(log_sigma)
nll = x_scaled + d * log_sigma + (1 + d) * kbe.log(1 + kbe.exp(-x_scaled))
```

c. Custom data normalization for MRIs

```python
import numpy as np
def encode_2_stage_normalization(images: np.ndarray, include_seg: bool = False) -> np.ndarray:
    # Apply the mask and return the images
    :param images: Input to be encoded
    :param include_seg: whether to include
    :return: Encoded input as numpy.ndarray
    
    data_src = np.squeeze(images[SRC_INDEX])
data_msk = np.squeeze(images[MSK_INDEX])
no_slices = data_src.shape[-1]
# scale each slice by respective bloodpool. If bloodpool unavailable, use closest one
bp_medians = np.array([])
for z in range(no_slices):
    bp_intensities = data_src[:, :, z][np.isclose(data_msk[:, :, z], const.GT_LABELS['lvbp'])]
    if np.sum(bp_intensities):
        bp_medians = np.append(bp_medians, np.median(bp_intensities))
    else:
        bp_medians = np.append(bp_medians, np.nan)
# Linearly interpolate to get median bp intensity for slices with no bp
nan_bp = np.isnan(bp_medians)
bp_medians[nan_bp] = np.interp(np.where(nan_bp)[0], np.where(np.logical_not(nan_bp))[0], bp_medians[np.logical_not(nan_bp)])
# Divide by twice the median bp to make that region ~.5
data_src_scaled = np.stack([data_src[:, :, z] / (2 * bp_medians[z]) for z in range(no_slices)], axis=-1)
# Zero out low intensities
min_viable = np.percentile(data_src_scaled[np.isclose(data_msk, const.GT_LABELS['lvmyo'])], 1)
data_src_scaled -= min_viable
```

117
data_src_scaled = np.clip(data_src_scaled, 0, np.inf)
data_src_scaled[np.logical_not(np.isclose(data_msk, const.GT_LABELS['lvmyo']))] = 0

if include_seg:
    data_src_scaled = np.stack([data_src_scaled, data_msk / const.GT_LABELS['lvmyo']], axis=-1)
else:
    data_src_scaled = np.expand_dims(data_src_scaled, axis=-1)

return data_src_scaled

d. Gradient activation map for regression

import os
import numpy as np
import keras
import keras.backend as kbe
import matplotlib.pyplot as plt
# crop is a custom cropping library

def viz_layer_ram():
    # Get model
    ancillary = False
    ensemble = False
    verbose = False
    model_path = os.path.join(data_config.VAL_RESULTS_SRC_PATH, '
        apnet_conv_loglogistic', 'trial_93')
    model_params = pickle.load(open(os.path.join(model_path, 'overall_model_params.pkl'), 'rb'))
    apnet_m, _ = compiled_model_from_params(model_params, ancillary, ensemble, verbose)
    apnet_m.load_weights(os.path.join(model_path, 'overall_apnet_model_weights.h5'), by_name=True)
    apnet_no_recon = Model(inputs=apnet_m.input, outputs=apnet_m.output[-1])
    apnet_m_flat = Sequential([apnet_no_recon.layers[0] + apnet_no_recon.layers[1].layers + [apnet_no_recon.layers[2]])

    last_conv_layer_name = 'activation_2'

    # Get data
    img_array_idx = 0
    train_id_to_get = 'P030'
    test_data = get_validation_data([train_id_to_get], verbose=0, ancillary=False, shuffle=False)
data_x_i = test_data[0]
    img_array = np.expand_dims(data_x_i[img_array_idx], axis=0)

    # Compute regression predictions
    mu_output = apnet_m_flat.output[:, 0] # first parameter is mode
last_conv_layer = apnet_m_flat.get_layer(last_conv_layer_name)

# This is the gradient of the top predicted class with regard to
# the output feature map of the last conv layer
grads = K.gradients(mu_output, last_conv_layer.get_output_at(-1))[0]

# This is a vector where each entry is the mean intensity of the gradient
# over a specific feature map channel
pooled_grads = K.mean(grads, axis=(0, 1, 2, 3))

iterate = K.function([apnet_m_flat.input], [pooled_grads, last_conv_layer.get_output_at(-1)[0]])

pooled_grads_value, conv_layer_output_value = iterate([img_array])

# We multiply each channel in the feature map array
# by "how important this channel is" with regard to the top predicted class
for i in range(pooled_grads.shape[-1]):
    conv_layer_output_value[:, :, :, i] *= pooled_grads_value[i]

# The channel-wise mean of the resulting feature map
# is our heatmap of class activation
heatmap = np.mean(conv_layer_output_value, axis=-1)

# Overlap the 2 images
idx = 2
plt.figure(1)
plt.imshow(data_x_i[img_array_idx][:, :, :, 0][:, :, idx])

plt.figure(2)
plt.imshow(data_x_i[img_array_idx][:, :, :, 1][:, :, idx])

# get high res version of image
rv = pickle.load(open(patient_path % train_id_to_get, 'rb'))
original_image = rv['_pixel_array']
original_seg = rv['_segmentation']

nx1, ny1, nz1 = original_image.shape
x1 = np.linspace(0, 1, nx1)
y1 = np.linspace(0, 1, ny1)
z1 = np.linspace(0, 1, nz1)
xv1, yv1, zv1 = np.meshgrid(x1, y1, z1)

nx2, ny2, nz2 = heatmap.shape
x2 = np.linspace(0, 1, nx2)
y2 = np.linspace(0, 1, ny2)
z2 = np.linspace(0, 1, nz2)

hmp_all = interp.interpn((x2, y2, z2), heatmap, (xv1, yv1, zv1), fill_value=0)
fov = original_seg
its = original_image
hmp_all_grads = hmp_all[np.isclose(fov, 4)]
reverse = False

# Scale the heatmap
m = np.min(hmp_all_grads)
M = np.max(hmp_all_grads)

if M <= 0:
a = .5 / (M - m)
b = -.5 * m / (M - m)
elif m >= 0:
a = .5 / (M - m)
b = (M - 2 * m) / (M - m)
elif np.abs(m) < np.abs(M):
a = .5 / M
b = .5
else:
a = -.5 / m
b = .5

hmp_all = np.clip(a * hmp_all + b, 0, 1)

its = 255 * its / np.max(its)

its_disp = its.copy()
its[np.isclose(fov, 4)] = 0  # 0 out area outside of ROI

bb = crop.find_bounding_box([its_disp[:, :, i] for i in range(its.shape[-1])],
force_square=True)
for i in range(its.shape[-1]):
im = crop.crop_image(its[:, :, i], bb)
im_disp = crop.crop_image(its_disp[:, :, i], bb)
hmp = crop.crop_image(hmp_all[:, :, i], bb)
fv = crop.crop_image(fov[:, :, i], bb)

im = keras.preprocessing.image.array_to_img(im, scale=True)
im_disp = keras.preprocessing.image.array_to_img(im_disp, scale=True)
im = keras.preprocessing.image.img_to_array(im)
im_disp = keras.preprocessing.image.img_to_array(im_disp)

# We rescale heatmap to a range 0-255
hmp = np.uint8(255 * hmp)

# We use jet colormap to colorize heatmap
jet = cm.get_cmap("RdBu")

# We use RGB values of the colormap
if reverse:
jet_colors = jet(np.arange(256)[::-1])[:, :3]
else:
jet_colors = jet(np.arange(256))[:, :3]
jet_heatmap = np.squeeze(jet_colors[hmp])

# We create an image with RGB colorized heatmap
jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)

# Superimpose the heatmap on original image
superimposed_img = jet_heatmap
superimposed_img[np.where(np.isclose(np.repeat(fv, 3, axis=-1), 0))] = 0
superimposed_img[np.where(np.isclose(np.repeat(fv, 3, axis=-1), 1))] = 0
superimposed_img += im
superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)
superimposed_img = superimposed_img.resize((1024, 1024))
superimposed_img.save('grad_cam/%d_3hm.png' % i)

im_disp = keras.preprocessing.image.array_to_img(im_disp)
im_disp = im_disp.resize((1024, 1024))
im_disp.save('grad_cam/%d_1img.png' % i)

r_map = fv * 0
r_map[np.isclose(fv, 1)] = 217
r_map[np.isclose(fv, 4)] = 27
g_map = fv * 0
g_map[np.isclose(fv, 1)] = 95
g_map[np.isclose(fv, 4)] = 158
b_map = fv * 0
b_map[np.isclose(fv, 1)] = 2
b_map[np.isclose(fv, 4)] = 119

seg_img = np.concatenate([r_map, g_map, b_map], axis=-1)
seg_img = keras.preprocessing.image.array_to_img(seg_img)
seg_img = seg_img.resize((1024, 1024))
seg_img.save('grad_cam/%d_2sg.png' % i)
References


5. Hinton G. Deep learning—a technology with the potential to transform health care. Jama 2018; 320:1101–2

7. Wang F, Casalino LP, and Khullar D. Deep learning in medicine—promise, progress, and challenges. JAMA internal medicine 2019; 179:293–4


43. Bickel PJ and Doksum KA. Mathematical statistics: basic ideas and selected topics, volumes I-II package. CRC Press, 2015


50. Zipes DP and Wellens HJJ. Sudden Cardiac Death. *Circulation* 1998; 98:2334–51. DOI: [https://doi.org/10.1161/01.CIR.98.21.2334](https://doi.org/10.1161/01.CIR.98.21.2334)


54. Wu KC. Sudden Cardiac Death Substrate Imaged by MRI: From Investigational Tool to Clinical Applications. Circ. Cardiovascular Imaging 2017 Jul; 10, 7. doi: https://doi.org/10.1161/CIRCIMAGING.116.005461


56. Scott PA, Morgan JM, Carroll N, Murday DC, Roberts PR, Peebles CR, Harden SP, and Curzen NP. The Extent of Left Ventricular Scar Quanti-
fied by Late Gadolinium Enhancement MRI Is Associated With Spontaneous Ventricular Arrhythmias in Patients With Coronary Artery Disease and Implantable Cardioverter-Defibrillators. Circulation: Arrhythmia and Electrophysiology 2011; 4:324–30. DOI: https://doi.org/10.1161/CIRCEP.110.959544


60. Mordi I, Jhund PS, Gardner RS, Payne J, Carrick D, Berry C, and Tzemos N. LGE and NT-proBNP identify low risk of death or arrhythmic events in patients with primary prevention ICDs. JACC: Cardiovascular Imaging 2014; 7:561–9


65. Arevalo HJ, Vadakkumpadan F, Guallar E, Jebb A, Malamas P, Wu KC, and Trayanova NA. Arrhythmia risk stratification of patients after myocardial infarction using personalized heart models. Nature Communications 2016 May; 7. doi: [https://doi.org/10.1038/ncomms11437](https://doi.org/10.1038/ncomms11437)


68. Zheng Q, Delingette H, Duchateau N, and Ayache N. 3D consistent and robust segmentation of cardiac images by deep learning with spatial propagation. IEEE Trans Med Imaging 2018


73. Zabihollahy F, Rajchl M, White JA, and Ukwatta E. Fully automated segmentation of left ventricular scar from 3D late gadolinium enhancement magnetic


86. Birsan T and Tiba D. One hundred years since the introduction of the set distance by Dimitrie Pompeiu. *IFIP Conference on System Modeling and Optimization*. Springer. 2005:35–9


Segmentation on Late Gadolinium Enhancement MRI: A Benchmark Study from Multi-Sequence Cardiac MR Segmentation Challenge. arXiv 2020


96. Ganesan AN, Gunton J, Nucifora G, McGavigan AD, and Selvanayagam JB. Impact of late gadolinium enhancement on mortality, sudden death and major adverse cardiovascular events in ischemic and nonischemic cardiomyopathy: a


110. Welch BL. The generalization of student’s’ problem when several different population variances are involved. Biometrika 1947; 34:28–35


Curriculum Vitae

Dan M. Popescu was born in Targoviste, Romania on August 9th, 1989. He graduated *summa cum laude* from University of Richmond, VA with a Bachelor of Science in Mathematics and a Bachelor of Arts in Economics, both with honors, in the Spring 2012. During his undergraduate education, Dan participated in research involving G-protein signal transduction, which he analyzed using Markov dynamics alongside his advisor, Dr. Ovidiu Lipan. After graduation, Dan joined Goldman Sachs & Co. in New York, NY, where he was a strategist in the Global Portfolio Solutions group, the institutional multi-asset class branch of Goldman Sachs Asset Management (GSAM). In 2016, he moved to Baltimore to start his Ph.D. in Johns Hopkins’ Applied Mathematics and Statistics (AMS) department. At Hopkins, Dan conducted his research alongside Dr. Natalia A. Trayanova and Dr. Mauro Maggioni. At Johns Hopkins, Dan was the recipient of the NIH T32 Training Fellowship for 2 consecutive years, the IDIES Seed Fund Fellowship for 1 year and the Rufus P. Isaacs Fellowship 4 years in a row.