

MSc in Computer Science 2017-18

Project Dissertation

Project Dissertation title:
Automated Feature Selection for Survival Analysis with Deep Learning

Term and year of submission: Trinity Term, 2018

Candidate Name: Carl Rietschel

Title of Degree the dissertation is being submitted under: MSc in Computer Science

Word count: 18,459 (determined by *texcount*)

Automated Feature Selection for Survival Analysis with Deep Learning



Candidate Number: 1020426

University of Oxford

A dissertation submitted for the degree of

MSc in Computer Science

August 2018

Acknowledgements

Working on this project has been an immensely rewarding experience for me, and I am extremely grateful for the support I have received along the way.

I would like to thank my supervisor Professor Mihaela van der Schaar for opening her inspiring research group to me and offering valuable advice and insights. Special thanks also goes to Jinsung Yoon for his expert guidance in our many fruitful discussions around feature selection, as well as Changhee Lee for introducing me to deep survival analysis and being so approachable for my countless questions. I finally cannot overstate the importance of my family and friends, whose encouragement and invaluable support I am so grateful to have, not just in this endeavour.

Abstract

Survival analysis has attracted the development of new deep learning models, most notably the recent state-of-the-art model DeepHit for competing risks. However, in real-world medical settings with many features and insufficient data, deep learning models can suffer from severe performance deficits.

In this dissertation, we propose robust changes to DeepHit's architecture and training procedure to counter this problem. Furthermore, we develop several methods to improve DeepHit's performance through novel approaches to automatic feature selection in deep survival analysis. We propose both filter and hybrid methods for hard feature selection, as well as neural network architecture adaptations that achieve soft feature selection.

Despite only sparse literature around feature selection for deep neural networks, and little previous evidence for its benefits, our experiments on two medical datasets demonstrate that substantial performance improvements against the original DeepHit model are achievable.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Motivation and objective 2

1.3 Overview of this dissertation 4

2 Background theory and literature **6**

2.1 Survival analysis 6

2.1.1 Basic statistical concepts 7

2.1.2 Censoring 9

2.1.3 Likelihood for censored data 10

2.1.4 Problem formulation 11

2.1.4.1 Survival data 11

2.1.4.2 Objective 11

2.1.5 Common survival models 12

2.1.5.1 Kaplan-Meier estimator 12

2.1.5.2 Cox proportional hazards 13

2.1.5.3 Fine-Gray 13

2.1.5.4 Random survival forest 14

2.2 Deep learning 15

2.2.1 Artificial neural networks 15

2.2.1.1 Artificial neuron 15

2.2.1.2 Deep feedforward networks 17

2.2.1.3 Training artificial neural networks 18

2.2.2 Deep learning approaches to survival analysis 20

2.3 Dimensionality reduction 21

2.3.1 Feature selection 22

2.3.1.1 Filter methods 22

2.3.1.2 Wrapper methods 24

2.3.1.3 Embedded methods 25
2.3.2 Feature selection in deep learning 25

3 Prediction task **29**
3.1 Medical relevance 29
3.2 Question for study 31
3.3 Description of PLCO dataset 31
3.4 Description of SEER dataset 33

4 Methodology **35**
4.1 DeepHit and DeepHit$^+$ 35
4.1.1 Cumulative incidence function estimation 35
4.1.2 Architecture 36
4.1.3 Loss function 37
4.1.4 Model settings and hyperparameters 37
4.1.5 DeepHit$^+$ improvements 38
4.2 Considerations for automatic feature selection 39
4.3 FilterDeepHit$^+$: Automatic feature selection using filter method . . . 39
4.3.1 Motivation 39
4.3.2 Modifications of DeepHit's architecture 40
4.3.3 Adaptation of filter methods to survival analysis with competing risks 41
4.3.4 Automatic feature selection 42
4.4 HybridDeepHit$^+$: Automatic feature selection using feature importance 43
4.4.1 Motivation 43
4.4.2 Feature importance 44
4.4.3 Automatic feature selection 46
4.5 SparseDeepHit$^+$: Automatic feature selection using sparse initial layer 46
4.5.1 Motivation 46
4.5.2 Architecture 47
4.6 AttentiveDeepHit$^+$: DeepHit with feature attention 48
4.6.1 Motivation 48
4.6.2 Architecture 48
4.6.3 Regularisation 50

5 Experiments **51**
5.1 Performance metrics 51
5.2 Experimental setting 52
5.2.1 Global settings 52
5.2.2 Benchmarks 53
5.2.3 DeepHit$^+$ 54
5.2.4 Extensions of DeepHit$^+$ 55
5.3 Performance results and analysis 57
5.3.1 Benchmarks, DeepHit and DeepHit$^+$ on PLCO 57
5.3.2 Automatic feature selection on PLCO 59
5.3.2.1 FilterDeepHit$^+$ 59
5.3.2.2 HybridDeepHit$^+$ 63
5.3.2.3 SparseDeepHit$^+$ 66
5.3.2.4 AttentiveDeepHit$^+$ 67
5.3.3 SEER results: The impact of a large feature set 68

6 Conclusion **71**

A Additional results **74**

Bibliography **77**

List of Figures

2.1 Structure of an artificial neuron 16
2.2 Structure of a deep feedforward neural network 17

3.1 PLCO: Distribution of survival times for each event 32
3.2 SEER: Distribution of survival times for each event 34

4.1 DeepHit architecture with two competing risks 36
4.2 ANOVA F-test p-values of PLCO features for Event 1 incidence at 120 months 40
4.3 Modified DeepHit$^+$ architecture with two competing risks 41
4.4 Method of calculating feature importance for a given trained DeepHit$^+$ model 44
4.5 SparseDeepHit$^+$ architecture with two competing risks 47
4.6 AttentiveDeepHit$^+$ architecture with two competing risks 49

List of Tables

5.1 Cumulative incidence of events by evaluation time for PLCO 52
5.2 Cumulative incidence of events by evaluation time for SEER 52
5.3 Random search hyperparameter choices for DeepHit 54
5.4 Random search hyperparameter choices for DeepHit$^+$ 55
5.5 Random search hyperparameter choices for the common hyperparameters in FilterDeepHit$^+$, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$ 55
5.6 Additional hyperparameter choices for FilterDeepHit$^+$ and HybridDeepHit$^+$ (M) and (P) 56
5.7 Additional hyperparameter choices for HybridDeepHit$^+$ (M/Cut) and (P/Cut) 56
5.8 Additional hyperparameter choices for SparseDeepHit$^+$ 56
5.9 Additional hyperparameter choices for AttentiveDeepHit$^+$ 57
5.10 C-index performance of benchmark models, DeepHit and DeepHit$^+$ on PLCO 57
5.11 Hyperparameters for DeepHit and DeepHit$^+$ on PLCO 59
5.12 C-index performance of all models on PLCO 60
5.13 Optimal hyperparameters for the FilterDeepHit$^+$ family on PLCO . . 61
5.14 Top 20 features by ANOVA p-value on PLCO 62
5.15 Top 20 features by SVM weights on PLCO 63
5.16 Top 20 features by ReliefF on PLCO 64
5.17 Optimal hyperparameters for the HybridDeepHit$^+$ family on PLCO . 64
5.18 Top 20 features by mean importance for DeepHit$^+$ on PLCO 65
5.19 Top 20 features by importance p-value for DeepHit$^+$ on PLCO 66
5.20 Top 20 features by first layer weights in SparseDeepHit$^+$'s sub-networks on PLCO 67
5.21 Optimal hyperparameters for SparseDeepHit$^+$ and AttentiveDeepHit$^+$ on PLCO 68

5.22 Top 20 features by average attention values in AttentiveDeepHit$^+$ on PLCO 69
5.23 C-index performance of DeepHit and DeepHit$^+$ on SEER and SEERsynth 70

A.1 Average C-index performance of all models across all evaluation horizons and events on PLCO 74
A.2 C-index performance of all models on SEERsynth 75
A.3 Average C-index performance of all models across all evaluation horizons and events on SEERsynth 76

Chapter 1

Introduction

1.1 Background

Survival analysis is a branch of statistics and machine learning that investigates time-to-events, i.e. the expected duration of time until one or more events of interest occur. Whilst widely applied in various fields, survival analysis is central to data-driven medical applications, and one objective is to learn from data how covariates (e.g. baseline patient information) affect time-dependent risks, such as life expectancy or the duration of freedom of symptoms for disease treatments (Cox and Oakes 1984). Survival analysis tasks may be distinguished by whether they consider a single risk, or several competing risks, where one of several events can occur.

Survival data is special due to a phenomenon called censoring that occurs when time-to-event observations are incomplete. For example, a medical study may conclude after 10 years, in which case patient deaths after this period are not recorded. The only information in the data will be that these patients did not experience any event for the duration of the study. As a consequence, classification models are not directly applicable to survival data, and specialized techniques need to be developed for this learning task.

Traditional survival analysis models include Kaplan and Meier's (1958) nonparametric population-level estimate of the survival distribution, i.e. the time distribution of a specific event. The first widely adopted model to include covariates is the single risk Cox proportional hazard model (also called Cox regression) (Cox 1972). The Cox model assumes a linear relationship between covariates and the logarithm of risk, a key model parameter: Risk is assumed to be proportional to the hazard function, which characterizes the survival distribution. The Fine-Gray model (Fine and Gray 1999) is an extension of Cox regression to the setting of competing risks, modelling the event-specific subdistribution hazard in a linear fashion. Another technique that

handles competing risks is using Random Survival Forests (RSF) (Ishwaran et al. 2008), which directly predict the event-specific cumulative incidence functions.

Deep artificial neural networks, machine learning models that have recently gained considerable popularity with groundbreaking achievements in many machine learning tasks (e.g. He et al. 2015; Silver et al. 2017), have been used in more modern survival analysis models. This began with extensions to the Cox model: Faraggi and Simon's (1995) model and DeepSurv (Katzman et al. 2016) use a single-layer and a multi-layer neural network, respectively, for more general feature-risk relationships, whilst Zhu, Yao and J. Huang (2016) use a convolutional neural network to deal with image data. However, drawbacks of the strong parametric assumptions of a proportional hazards model remain in all these Cox-based approaches.

Most recently, the model DeepHit (Lee et al. 2018) was developed in order to overcome this challenge by using a neural network to learn the survival distribution directly. It avoids the difficulties of parametric methods by making no assumptions about the underlying stochastic process, and allows for the possibility that the relationships between features and risks change over time. Furthermore, it works naturally in the competing risks setting by adopting a multitask learning approach for each event-specific survival function. DeepHit has been shown on the datasets in (Lee et al. 2018) to achieve large performance improvements over previous methods, including Cox and Fine-Gray.

In this work, we focus on the application of survival analysis to a real-world task of medical relevance using data from the US-based Prostate, Lung, Colorectal and Ovarian Cancer Screening Trial (PLCO). We predict prostate cancer incidence from patient screening data and consider competing risks, with death prior to prostate cancer incidence the secondary event of interest. Medical benefits include improved personalised cancer predictions, which in addition to their informational value could be used to combat overdiagnosis and overtreatment, a common problem in prostate cancer (Pashayan, Pharoah et al. 2015). In an ancillary experiment on a modified dataset from the Surveillance, Epidemiology, and End Results Program (SEER), we analyse the impact of feature selection for DeepHit-based models.

1.2 Motivation and objective

Whilst the deep learning model DeepHit may outperform traditional methods on some datasets (Lee et al. 2018), we find that immediate application of DeepHit on the PLCO dataset shows significant performance deficits: In common with deep learning

methods in many applications, in real-world medical settings such as PLCO with a large number of features and insufficient number of samples, DeepHit can lose significant accuracy.

The aim of this work is to improve DeepHit to overcome these shortcomings in two ways:

1. As enabling groundwork, we improve DeepHit's architecture and training procedure to be more tailored to varied competing risks situations.

2. We then develop novel deep learning survival models based on DeepHit that apply automatic feature selection techniques to improve their performance.

Initial architecture and training procedure improvements We develop architectural improvements to DeepHit as pre-requisites for the abovementioned feature selection models. We present the resulting model as DeepHit$^+$, which itself aims to achieve significant performance improvements over DeepHit's original implementation. Firstly, DeepHit$^+$ allows adaptive and event-specific sizing of the neural network in a competing risks setting. It then uses random search (Bergstra and Bengio 2012) to find the optimal architecture for a given dataset. Finally, we improve the network's early stopping procedure (e.g. Goodfellow, Bengio and Courville 2016) in training.

Automatic feature selection Reducing the dimension of the input feature space is a common technique in machine learning. Feature selection is a way of dimensionality reduction that preserves the original semantics of the data and thus supports model interpretability, an important consideration for medical models. Whilst for many machine learning methods feature selection is standard, there has been reduced focus on feature selection for deep learning models due to their prowess at feature engineering, that is automatically learning suitable data representations. Literature around feature selection and evidence of significant benefits for deep learning models is sparse, and to the best of our knowledge we are the first to apply it to deep survival analysis.

We distinguish hard and soft feature selection; the former eliminates features entirely, whilst the latter reduces their weights in the model. For each we develop and test several novel models:

- The family of FilterDeepHit$^+$ methods uses model-independent hard feature selection methods as a pre-processing step. To this end, we adapt conventional methods for feature selection to work in the special context of survival analysis.
- The family of HybridDeepHit$^+$ methods performs hard feature selection that is integrated into the model, and thus can account for model-feature interactions. The methods are inspired by techniques to extract feature importance values from trained machine learning models.
- In SparseDeepHit$^+$, we introduce additional specially crafted layers to the model inputs, inspired by Li, C.-Y. Chen and Wasserman (2016). These perform soft feature selection, reducing weights on less relevant features.
- AttentiveDeepHit$^+$ achieves soft feature selection through a self-learnt sub-network that reweights the input features.

We implement all the above methods in the context of survival analysis with competing risks, and show that substantial performance improvements are achievable. Whilst the gains we show are specific to survival analysis on certain datasets, feature selection approaches may benefit and stimulate further research in other deep learning applications, such as models for classification or regression problems.

1.3 Overview of this dissertation

This dissertation consists of six chapters.

- Chapter 1 gives a short introduction of this project and the dissertation structure.
- In Chapter 2, we present the necessary theoretical background that we use in this dissertation and review related literature. In particular, we first introduce survival analysis, giving basic definitions of the main statistical concepts as well as censoring, and providing the mathematical formulation for the survival analysis problem we address. We also define and explain the common traditional survival analysis models. In the second section, we introduce deep feedforward neural networks, techniques for how they are trained, as well as existing deep learning approaches to survival analysis. The final section of Chapter 2 defines dimensionality reduction and feature selection, and presents an overview of currently used methods. We then review in depth the existing literature on feature selection in deep learning.

- In Chapter 3, we introduce the specific prediction task and datasets we use in this work. We discuss the relevance of our application in prostate cancer research, and precisely define the prediction task. This is followed by a presentation of the PLCO dataset, on which our main work is based, as well as a detailed analysis of its characteristics. Finally, we also exhibit details of the Surveillance, Epidemiology, and End Results Program (SEER) dataset, which we use in parts of this work to further illustrate the benefits of our models.

- In Chapter 4, we outline details of the machine learning methods that we develop in this dissertation. Beginning with a review of DeepHit's architecture, we explain the improvements we made for the model DeepHit$^+$. We then give motivations for, and details of each of the new automatic feature selection models that we propose. For FilterDeepHit$^+$ we outline a method to adapt traditional feature ranking techniques to survival analysis and outline a procedure for the neural network to utilize these. For HybridDeepHit$^+$ we present a way to extract feature importances from a deep learning model and apply them for feature selection in an integrated fashion. Finally, we explain the architectures of both SparseDeepHit$^+$ and AttentiveDeepHit$^+$ and how they introduce soft feature selection.

- Chapter 5 presents the experimental analysis and discussion of our models. We begin with defining the C-index metric (Gerds et al. 2013) that we use to benchmark model performance. This is followed by technical details of the model implementations, including evaluation times and how we construct train-test data splits. For each model, we also give the (hyper-)parameter settings used for obtaining our results. The main body of the Chapter exhibits and discusses the performance of the individual models. We give comparisons between DeepHit and DeepHit$^+$, as well as detailed performance gains attained by FilterDeepHit$^+$, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$. We also show how all methods compare to benchmarks of traditional survival analysis models Cox, Fine-Gray and RSF.

- We conclude the dissertation in Chapter 6 with a critical summary of our work and its contributions to both machine learning and medical research, as well as presenting possible approaches to further research on this topic.

Chapter 2

Background theory and literature

In this chapter we introduce the technical concepts underlying our work, and review related literature. The first section presents survival analysis, including the main mathematical notation we use in this dissertation and the most important traditional survival analysis models. We subsequently give an introduction to deep learning, and review the modern approaches to survival analysis using neural networks. In the final section we explain dimensionality reduction and its motivations, giving an overview of the most common feature selection methods and reviewing approaches in the literature around feature selection for deep learning.

2.1 Survival analysis

Survival analysis (see e.g. Klein and Moeschberger 2003), also called *time to event analysis*, is a collection of methods in statistics and machine learning that is frequently used in economics (e.g. Bhattacharjee et al. 2007; Danacica and Babucea 2010), finance (e.g. Bellotti and Crook 2009; Giot and Schwienbacher 2007), engineering (e.g. Eyal et al. 2014; Lawless, Hu and Cao 1995) and medicine (e.g. Cox and Oakes 1984; Lawless 1982; Pfirrmann et al. 2015; Rezaianzadeh et al. 2009). The objective of survival analysis is to understand the duration of time until one or more events happen. In particular, we wish to learn the relationships between covariates (such as baseline patient data) and the first hitting time of a specific event (e.g. death, disease onset, etc.). When there is more than one event of interest we speak of competing risks, e.g. when analysing whether a patient will develop prostate cancer during his lifetime, or die first.

We focus our work on the medical domain, where clinicians may use survival analysis to make screening or treatment decisions. Throughout this work, we therefore often use medical language to describe survival analysis concepts: Training or test

instances/examples are referred to as *patients*, the first hitting time as *survival time*, and we may refer to events as *causes*. Whilst the specific prediction task for this dissertation (outlined in Chapter 3) is in medicine, we emphasize that all models we develop are applicable for survival analysis in any given field.

2.1.1 Basic statistical concepts

Formally, we consider a set of K events, denoted $\{E_k : k = 1, 2, \ldots, K\}$. This gives rise to the following distinction between two sub-fields of survival analysis:

- *Single risk*, where there is only one event of interest ($K = 1$)
- *Competing risks*, where there are several events of interest ($K \geq 2$). In this work, we assume exactly one event can occur for each patient (e.g. a patient eventually dies, but only from one specific cause (Gooley et al. 1999)). There is a more general case in which several events, such as the onsets of various diseases, may occur.

We continue outlining the theory for general K, noting differences between the single risk and competing risks setting when required. We consider the *time-to-event* random variable T, which we assume to be non-negative. This is measured from a pre-defined start event, which in medical cases can be trial entry or hospital admission, until event incidence, e.g. disease onset or death. We further define an *event indicator* δ to denote which event was observed at time T.

The scale in which T is measured need not be time in the strict sense, such as in (Eyal et al. 2014) where mileage is used to quantify the time until automobile components failure. However, in the medical setting, an actual time scale is most commonly applicable, and we shall focus on that in this work.

In competing risks settings, our interest often lies in the probabilities that summarize our knowledge of the likelihood that a certain competing risk will happen. Klein and Moeschberger (2003) present three different types of probabilities one may consider. In the single risk setting, these concepts collapse into one.

- *Crude probabilities*: The probability of death from a specific cause in the real world where also all other risks are acting on the individual. In this work, our focus is on estimating crude probabilities.
- *Net probabilities*: The probability of death from a specific cause in a hypothetical world where only this one risk is acting on the individual (e.g. the chance a

man will die of prostate cancer in the counterfactual world where one can only die of prostate cancer).

- *Partial crude probabilities*: The probability of death from a specific cause where some risks of death are eliminated.

Cumulative incidence function The cause-specific *cumulative incidence function* (CIF) is a central concept to survival analysis. For each event E_k the CIF gives the crude probability that this event occurs on or before time time t, and is denoted $F_k(t)$. As in (Fine and Gray 1999), we therefore define it as

$$F_k(t) = \mathbb{P}\left[T \leq t, \delta = k\right]. \tag{2.1}$$

The survival probability up to a time t, also called *survival function* and denoted $S(t)$, is consequently defined as

$$S(t) = 1 - \sum_{k=1}^{K} F_k(t) = \mathbb{P}\left[T > t\right]. \tag{2.2}$$

We often assume that the time distributions are continuous. In this case we have a *probability density function* $f_k(t)$, and

$$F_k(t) = \int_0^t f_k(s)ds, \tag{2.3}$$

whilst for discrete random variables we suppose t can take values on $t_j, j = 1, 2, \ldots$ and have a *probability mass function* $f_k(t_j)$ such that

$$F_k(t) = \sum_{t_j <= t} f_k(t_j). \tag{2.4}$$

The hazard function In survival analysis we also commonly consider a quantity called the *cause-specific hazard rate* $h_k(t)$, also known as the *force of mortality*. It represents the instantaneous risk of event incidence (e.g. dying) for those still at risk (e.g. patients still alive). It is defined by

$$h_k(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}\left[t \leq T < t + \Delta t, \delta = k \mid T \geq t\right]}{\Delta t}, \tag{2.5}$$

and is used by the Cox proportional hazards models (see Section 2.1.5.2) and its variants.

2.1.2 Censoring

The special nature of survival data is to a large part due to a phenomenon called *censoring*. This occurs when lifetimes are known to have occurred only within certain intervals. For example, in clinical trials it is not always possible to wait until a patient develops a certain disease, or the patient may never develop the disease. Censored data is still useful for survival analysis, as there is valuable information in the fact that a patient did not experience an event over a certain timeframe. There are various types of censoring, determined by the method used to obtain the survival data, defined by Klein and Moeschberger (2003) as follows.

Right censoring Observations are right-censored if the event is observed only if it occurs prior to a certain time. For example, resource and time constraints may mean a medical trial has a limited time span. When the trial ends, patients who have not experienced any event are right-censored.

In this case, we define the *censoring time* as the random variable C_r, the *observed time* Y as

$$Y = \min(T, C_r), \tag{2.6}$$

and adapt the event indicator δ to show whether an event was observed: $\delta = 0$ for censored observations, and $\delta = k$ when event E_k was observed. We distinguish between three types of right-censoring:

1. *Type 1 censoring*: This occurs when an experiment stops at a predetermined time, at which all remaining subjects are right-censored.

2. *Type 2 censoring*: Here the experiment stops after a predetermined number are observed to have experienced the event of interest.

3. *Random censoring*: In this case, some individuals may experience a competing event that is treated as censoring, such as migration out of the study center or accidental death.

Left censoring We speak of left censoring if the event time cannot be observed before a censoring time C_l. For such cases it is known which event they have experienced at some time point before C_l, but not exactly when. For example, a survey question about when an individual started smoking may include the option "I have smoked, but cannot remember exactly when the first time was". A participant choosing this

response corresponds to a left-censored event time: It is only known that they started smoking on or before their current age. The observed time is thus defined as

$$Y = \max(T, C), \tag{2.7}$$

and in addition to event indicator δ (for competing risks), *censoring indicator* ϵ shows whether the exact survival time is observed ($\epsilon = 1$) or not ($\epsilon = 0$).

Double censoring When both left censoring and right censoring occur, we consider survival times as doubly censored. In the above example of a smoking survey the response "I have never smoked" leads to additional right-censoring, and thus this is a doubly censored sampling scheme. We therefore observe

$$Y = \max(\min(T, C_r), C_l). \tag{2.8}$$

Interval censoring In certain circumstances, the survival time is only known to occur within an interval. An example for interval censoring may be in a clinical trial that only has periodic disease diagnostic checkups. Instead of observing a single event time, we observe

$$Y = (L, R], \tag{2.9}$$

with L the left and R the right endpoint of the censoring interval.

2.1.3 Likelihood for censored data

Parametric and semiparametric methods for survival analysis require a likelihood function that is optimized during training of the model. Each data point yields different information, depending on the type of censoring, and this is reflected in its construction. In order to proceed, we must make the assumption that survival time and censoring time are independent (Leung, Elashoff and Afifi 1997). We shall keep this assumption throughout this work.

The information contained in a data point where the exact survival time (and corresponding event E_k) is known is the probability density $f_k(T)$. For a right censored observation we know survival up to the censoring time, encapsulated in $S(C_r)$. Similarly, left-censored observations contribute $F_k(C_l)$. For interval censored data the information is the probability that the event time is in this interval, i.e. $F_k(R) - F_k(L)$.

Thus, based on Klein and Moeschberger (2003) and extended to incorporate competing risks, the likelihood function is

$$L \propto \prod_{i \in E} f_{k^{(i)}}(T^{(i)}) \prod_{i \in R} S(C_r^{(i)}) \prod_{i \in L} F_{k^{(i)}}(C_l^{(i)}) \prod_{i \in I} \left[F_{k^{(i)}}\left(R^{(i)}\right) - F_{k^{(i)}}\left(L^{(i)}\right) \right], \tag{2.10}$$

where E denotes the set of individuals i with known exact survival times $T^{(i)}$ and event $k^{(i)}$, and R, L and I denote the set of right, left and interval censored individuals, respectively.

2.1.4 Problem formulation

In this section we present the structure of the datasets for our survival models. Henceforth we shall focus on right censored data, the most common type of censoring (Kartsonaki 2016) and frequently encountered in time-limited medical trials.

2.1.4.1 Survival data

Survival data gives three pieces of information for every patient/instance, a tuple $(\mathbf{x}, \tau, k)$:

1. *Covariates* $\mathbf{x}$ are the baseline characteristics of each patient observed (in machine learning literature also called *features*). The covariate data for a patient is assumed to be numerical and is a length D vector, where D is the number of features. Categorical features are assumed to be encoded appropriately into numerical components of $\mathbf{x}$, e.g. using one-hot encoding (Murphy 2012).

2. *Time* τ observed from when the covariates were collected until the first event or censoring. As in Equation 2.6, $\tau = \min(T, C)$, where T and C are the event and censoring times, respectively.

3. *Label* $k \in \{\emptyset, 1, \ldots, K\}$ indicating which event (or right censoring, which is denoted $\emptyset$) occurred at time τ.

Our dataset is thus

$$\mathcal{D} = \left\{(\tau^{(i)}, k^{(i)}, \mathbf{x}^{(i)})\right\}_{i=1}^{N}, \tag{2.11}$$

describing observations for N patients.

2.1.4.2 Objective

A machine learning or statistical model will attempt to use the data to give empirical estimates $\hat{F}_k$ of the true cumulative incidence functions F_k. The objective is not solely to learn the general incidence distribution for the population, but rather to understand the relationship between the distribution of hitting times and patient covariates. We therefore allow the cumulative incidence functions to depend on the

covariates $\mathbf{x}$, as well as the time point t for which to predict. The true cumulative incidence function thus becomes $F_k(t \mid \mathbf{x})$, and its estimate $\hat{F}_k(t \mid \mathbf{x})$.

We wish to obtain close estimates of the true functions, and therefore consider performance metrics to benchmark how well a trained survival model explains test data. Further details are given in Section 5.1.

2.1.5 Common survival models

There is a wealth of previous literature on techniques that have been crafted to deal with censored data. In this section we outline the currently most widely used models. A drawback of many current methods is the necessity to assume a specific parametric form for the underlying stochastic distributions. Such methods approach the problem of survival analysis by learning the parameters from available data. Through their strong parametric assumptions they may be severely limited in their ability to capture general relationships between covariates and survival distributions (Lee et al. 2018). In the four models in this section, Random Survival Forests (Ishwaran et al. 2008) are the only nonparametric approach, whilst later in Section 2.2.2 we present newer deep learning survival models, including the state-of-the-art nonparametric approach DeepHit.

2.1.5.1 Kaplan-Meier estimator

The *Kaplan-Meier estimator* (Kaplan and Meier 1958) is one of the most widely used survival models in the statistical and medical research literature. It is nonparametric and thus very flexible in learning survival curves, however at the disadvantage of not incorporating patients' covariates. The model may thus be used at the population level, but not for individualised risk predictions.

Given a dataset $\mathcal{D}$ with competing risks as defined in Section 2.1.4.1, the estimator of the overall survival curve is given by Houwelingen and Putter (2011) as

$$\hat{S}(t) = \prod_{\tau^{(i)} \leq t} \left(1 - \frac{\mathbf{1}_{\{\delta^{(i)}>0\}}}{Y(\tau^{(i)})}\right), \tag{2.12}$$

where $Y(t)$ is the size of the risk set $R(t)$ of patients still at risk just prior to t, i.e. $R(t) = \{i : \tau^{(i)} \geq t\}$. The estimators for the cumulative incidence functions are then of the form

$$\hat{F}_k(t) = \sum_{\tau^{(i)} \leq t} \frac{\mathbf{1}_{\{\delta^{(i)}=k\}}}{Y(\tau^{(i)})} \cdot \hat{S}(t^{(i)}-), \tag{2.13}$$

where $t-$ denotes the time just prior to t.

2.1.5.2 Cox proportional hazards

The *Cox proportional hazard* model (or *Cox model*) (Cox 1972) is an early and one of the most widely known and used survival models that includes patient covariates. By construction, the Cox model is only able to deal with one event of interest rather than multiple competing risks. It considers the individual specific hazard function $h(t)$ dependent on covariates $\mathbf{x}$ (we drop the individual superscript $^{(i)}$ for notational convenience) and makes the *proportional hazards assumption* (Houwelingen and Putter 2011) that

$$h(t) = \exp(\mathbf{x}^\top \beta) h_0(t). \tag{2.14}$$

Here, $h_0(t)$ is called the *baseline hazard* that determines the shape of the survival functions and is commonly estimated nonparametrically, and β are the model parameters to be learned. It follows that the survival function for general covariates $\mathbf{x}$ is given by

$$S(t \mid \mathbf{x}) = S_0(t)^{\exp(\mathbf{x}^\top \beta)}. \tag{2.15}$$

The strong parametric assumptions lie in the fact that each individual's hazard function must be proportional to the baseline hazard for all t. Furthermore, the log of the individual hazard rate is a linear function of the covariates. This means that the model does not cater for non-linear relationships between covariates and risk, or varying relationships over time.

As Cox models only consider single risks, they are not directly applicable to a competing risks setting. In principle, event-specific models may be applied to the competing risks setting by focusing on one event at a time, and treating all others as censoring. However, this approach is only appropriate when the competing risks are independent, which may often not hold (Lee et al. 2018).

2.1.5.3 Fine-Gray

In contrast to the Cox model, the model proposed by Fine and Gray (1999) is specifically designed for the competing risks setting by not treating competing events in the same way as censored observations. Instead, their method estimates a proportional hazards model for a quantity called the *subdistribution hazard*.

The subdistribution hazard $h_k^*(t)$, originally described by Gray (1988), is given by

$$h_k^*(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}\left[t \leq T < t + \Delta t, \delta = k \mid (T \geq t) \cup (T \leq t \cap \delta \neq k)\right]}{\Delta t}. \tag{2.16}$$

It is similar to a cause-specific hazard, with one difference: Whilst the cause-specific hazard removes individuals from the risk set when any event occurs, the subdistribution hazard only removes an individual when the specific event of interest or true censoring occurs. This corresponds to modelling the hazard function for the improper random variable $\tau_k^* = \tau \cdot \mathbf{1}_{\{\delta=k\}} + \infty \cdot \mathbf{1}_{\{\delta \neq k\}}$. The implied distribution function is equal to $F_k, t < \infty$, and a point mass at $t = \infty$ with $\mathbb{P}[\tau_k^* = \infty \mid \mathbf{x}] = \mathbb{P}[\tau < \infty, \delta \neq k \mid \mathbf{x}] = 1 - F_k(\infty \mid \mathbf{x})$. The Fine-Gray model can therefore be seen as modelling the cumulative incidence function directly, and is given by

$$h_k^*(t) = \exp(\mathbf{x}^\top \beta_k) h_{0k}^*(t), \tag{2.17}$$

where β_k is a model parameter, and $h_{0k}^*(t)$ is the baseline subdistribution hazard that is estimated nonparametrically. The cumulative incidence function is then

$$F_k(t \mid \mathbf{x}) = 1 - \exp\left(-\int_0^t \exp(\mathbf{x}^\top \beta_k) h_{0k}^*(s) ds\right). \tag{2.18}$$

Whilst incorporating competing risks, the Fine-Gray model has limitations similar to Cox models by its strong assumptions on the hazard rates and the linear way in which covariates effect parameters (Lee et al. 2018).

2.1.5.4 Random survival forest

Random Survival Forests (RSF), originally proposed by Ishwaran et al. (2008) and adapted for competing risks in (Ishwaran et al. 2014), are an ensemble tree method for survival analysis that extend Random Forests (Breiman 2001). They use the special characteristic of decision trees that instead of having a terminal node correspond to a certain value (e.g. a class label for classification tasks), other outputs can be specified. In particular, in a RSF a terminal node yields estimates of cause-specific cumulative incidence functions.

A RSF is a collection of randomly grown *survival trees*, each of which is grown using an independent *bootstrap sample* (a smaller sample drawn with replacement) of the learning data using random feature selection at each node. The general approach is to grow a single competing risk tree in each bootstrap sample, as shown in the competing risks forest algorithm (from Ishwaran et al. 2014):

1. Draw B bootstrap samples from the learning data.

2. Grow a competing risk tree for each bootstrap sample. At each node, randomly select $M \leq p$ candidate variables. Split the node using the candidate variable that maximizes a competing risk splitting rule: A generalized log-rank test or a Gray's test.

3. Grow the tree to full size under the constraint that a terminal node should have no less than $n_0 > 0$ unique cases.

4. For estimation given covariates $\mathbf{x}$, obtain the final node of each tree corresponding to $\mathbf{x}$, and calculate event-specific cumulative incidence estimators for these nodes. Take the average over all B trees to obtain the ensemble estimates.

As outlined by Ishwaran et al. (2008), Random Forests have been shown to have highly accurate performance, comparable to methods such as *bagging*, *boosting* and *support vector machines*. Random Survival Forests therefore have various advantages over previous methods (Ishwaran et al. 2014), including direct estimation of the cumulative incidence function, ability to capture non-linear feature-label relationships, effectiveness in high-dimensional settings and incorporation of competing risks, and they have been shown to outperform Cox on empirical datasets.

2.2 Deep learning

Deep learning is a family of machine learning methods loosely inspired by the structure of the human brain. Due to their flexibility and power, many recent successes in artificial intelligence are based upon deep learning techniques, such as the mastering the game of Go using the computer program *AlphaGo* (Silver et al. 2017) or beating human performance in image classification (He et al. 2015).

Deep learning methods use *artificial neural networks* to learn from data, and we outline their general workings below. Later in this section we review current deep learning approaches to survival analysis.

2.2.1 Artificial neural networks

2.2.1.1 Artificial neuron

The first artificial neural network (Rosenblatt 1958) was called the *perceptron* and consisted of a single *artificial neuron*. The perceptron transforms binary inputs $x_1, \ldots, x_m$ into a single binary output y:

$$y = \begin{cases} 0, & \text{if } \sum_{i=1}^{m} w_i x_i + b \leq 0 \\ 1, & \text{if } \sum_{i=1}^{m} w_i x_i + b > 0, \end{cases} \tag{2.19}$$

where *weights* w_i and bias b are the model parameters.

An artificial neuron used in current neural networks follows a similar structure, extended to real-valued inputs and with a general non-linear *activation function* f

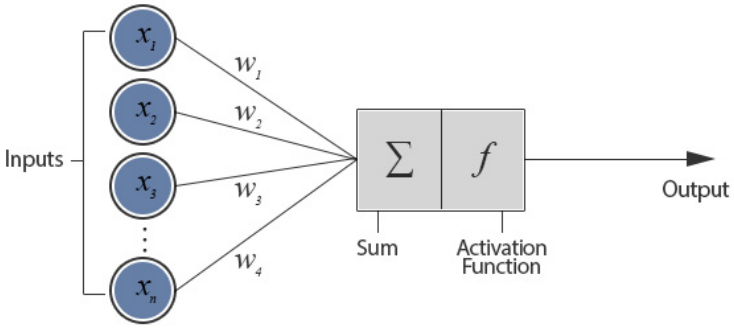


Figure 2.1: Structure of an artificial neuron

replacing the step function in Equation 2.19. With inputs $\mathbf{x}$ and weights $\mathbf{w}$ now written in vector form, the output y is illustrated in Figure 2.1 (from Jacobson 2013) and given by

$$y = f(\mathbf{w}^\top \mathbf{x} + b). \tag{2.20}$$

There are various choices of activation functions, and their design is an extremely active research area (Goodfellow, Bengio and Courville 2016). Current commonly used functions are outlined below.

Rectified linear units (ReLU) These are defined by

$$f(z) = \max(0, z). \tag{2.21}$$

The ReLU activation function is common default choice due to various favourable properties (Goodfellow, Bengio and Courville 2016): They are easy to optimize due to their linearity and have a large gradient wherever they are active to sidestep the *vanishing gradient problem* that one may encounter with other choices of activation function (Goh, Hodas and Vishnu 2017). However, one drawback is that they cannot learn via gradient-based methods on examples for which their activation is zero.

There are various generalizations of ReLUs of the form

$$f(z) = \max(0, z) + \alpha \min(0, z) \tag{2.22}$$

that achieve a nonzero slope when $z < 0$. *Absolute value rectification* fixes $\alpha = -1$, a *leaky ReLU* (Maas 2013) sets α to a small value such as 0.01 and *parametric ReLUs* (He et al. 2015) treat α as a learnable parameter.

Logistic sigmoid and hyperbolic tangent Before ReLUs became widespread, most neural networks used the logistic sigmoid activation,

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.23}$$

or the hyperbolic tangent,

$$f(z) = \tanh(z) = 2\sigma(2z) - 1. \tag{2.24}$$

Unlike ReLUs, these activation functions *saturate* with vanishing gradient when their input z becomes a very high or very low value, impeding gradient-based learning (Goodfellow, Bengio and Courville 2016). Their use is therefore now discouraged.

2.2.1.2 Deep feedforward networks

A deep neural network is a more complex model composed by connecting more layers than one, each containing several artificial neurons. The first layer is often called the *input layer*, whilst the final layer is called the *output layer*. Middle layers are called *hidden layers*. The prerequisite for a *feedforward* neural network is that the connections between neurons create no cycles. Often we speak of *fully connected* layers, meaning that all neurons from the previous layer serve as inputs to all of the current layer's neurons, as shown in the example in Figure 2.2 (from Jacobson 2013). Feedforward neural networks are often characterised by their number of layers and the number of neurons (or *nodes*) in each layer.

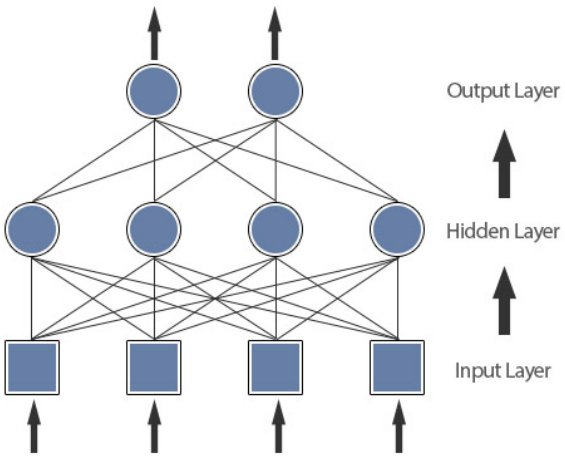


Figure 2.2: Structure of a deep feedforward neural network

Forward equations Now that we have several artificial neurons in each layer, let z^l denote the values of the neurons (called *pre-activations*) prior to application of the activation function f^l. Furthermore, we denote the outputs (or *activations*) of the lth layer as $\mathbf{a}^l$. We now have a matrix of weights $\mathbf{W}^l$ and a vector of biases $\mathbf{b}^l$. A fully connected layer can thus be characterized by the equations

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l \tag{2.25}$$

$$\mathbf{a}^l = f^l(\mathbf{z}^l). \tag{2.26}$$

Output units Often the activation function for hidden layers is applied individually for each neuron. This may differ for output units, as their role is to provide some additional transformation to complete the task desired of the network. Common choices of output activation functions are:

- *Linear units* with activation $f(\mathbf{z}) = \mathbf{z}$, often used when the output can be any real value, e.g. modelling the mean of a conditional Gaussian distribution.
- *Sigmoid units* with activation $f(\mathbf{z})_i = \sigma(z_i)$ as in Equation 2.23, often used when the output is a scalar and should represent a probability in $(0, 1)$.
- *Softmax* with activation $f(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$, used when the output vector is required to be a probability distribution whose components sum to 1.

2.2.1.3 Training artificial neural networks

Loss functions Artificial neural networks are most commonly trained via minimisation of a *loss* or *cost* function designed to capture how well model output fits the training data. The quantity of choice is often the negative log-likelihood, which depends on the specific prediction task at hand: In Section 2.1.3 we have seen likelihood functions that may serve as loss function for survival models. The deep learning survival model DeepHit uses such a function supplemented by a second specifically designed loss function, as shown in Section 4.1.

For regression tasks with real-valued outputs, a common loss function is the *mean squared error* (MSE) that corresponds to maximum likelihood assuming a Gaussian underlying distribution (Goodfellow, Bengio and Courville 2016). On a single datapoint $(\mathbf{x}_i, y_i)$ with prediction $\hat{y}_i$ it is defined by

$$\ell_i = \|y_i - \hat{y}_i\|_2^2 . \tag{2.27}$$

For classification tasks the *cross entropy* loss function is commonly used, given by

$$\ell_i = -(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)). \tag{2.28}$$

The full loss function $\mathcal{L}$ on a dataset of size N is built up of the sum of the loss for all individual datapoints, i.e. $\mathcal{L} = \frac{1}{N} \sum_i \ell_i$.

Gradient descent Loss function minimisation can be done by *gradient descent*, i.e. iterative updating of model parameters θ using

$$\theta_{t+1} = \theta_t - \eta_t \mathbf{g}_t = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t). \tag{2.29}$$

Here η_t is called the *learning rate*, and must be fixed before training, whilst the initial parameters θ_0 are set using a specific *initialization* technique. Since calculating the gradient $\nabla \mathcal{L}(\theta_t)$ can be very costly on large datasets, one often calculates this gradient on a random subsample of the training dataset called a *minibatch*, which is an unbiased estimator of the original gradient (Goodfellow, Bengio and Courville 2016). This technique is called *stochastic gradient descent* (SGD), and has the advantage that computation time per update does not grow with the number of training examples.

Backpropagation In order to calculate the gradients of the neural network, we apply a technique called backpropagation (Rumelhart, Hinton and Williams 1986). Based on a single datapoint's loss ℓ, L layers, and the notation of the *forward equations* from Equations 2.25 and 2.26, the backpropagation equations are:

1. Compute
$$\frac{\partial \ell}{\partial \mathbf{z}^L} = \frac{\partial \ell}{\partial \mathbf{a}^L} \cdot \frac{\partial \mathbf{a}^L}{\partial \mathbf{z}^L} \tag{2.30}$$
2. For $l = L - 1, \ldots, 1$ calculate
$$\frac{\partial \ell}{\partial \mathbf{z}^l} = \frac{\partial \ell}{\partial \mathbf{z}^{l+1}} \cdot \mathbf{W}^{l+1} \cdot \frac{\partial \mathbf{a}^l}{\partial \mathbf{z}^l} \tag{2.31}$$
3. This gives the derivatives for the model parameters as
$$\frac{\partial \ell}{\partial \mathbf{W}^l} = \left(\mathbf{a}^{l-1} \frac{\partial \ell}{\partial \mathbf{z}^l} \right), \text{ and} \tag{2.32}$$
$$\frac{\partial \ell}{\partial \mathbf{b}^l} = \frac{\partial \ell}{\partial \mathbf{z}^l}. \tag{2.33}$$

Additional techniques There are various additional more sophisticated techniques that are common practice in deep learning research, and that we use to train the neural networks in this dissertation. We briefly review these here:

Regularisation techniques are modifications that intend to reduce a model's generalization error but not its training error (Goodfellow, Bengio and Courville 2016). Very common is adding parameter norm penalties to the loss function, such as L_1 *regularisation* using the absolute value, and L_2 *regularisation* using the square of each parameter. *Adam* (Kingma and Ba 2014) is a stochastic gradient descent optimisation algorithm with adaptive learning rates that has recently become commonly used since it is insensitive to hyperparameters (Goodfellow, Bengio and Courville 2016). Furthermore, many neural networks apply *dropout* (Srivastava et al. 2014) as a computationally inexpensive method for regularisation, introducing an additional hyperparameter called the keep probability. Finally, the *Xavier initializer* (Glorot and Bengio 2010) is a popular choice to automatically set the scale of initialized weights and biases based on the network architecture.

When hyperparameters, such as the number and dimension of layers, are not picked manually, often *grid search* across the entire hyperparameter space is used. This finds the best model using performance evaluated on a *validation set* that is separate from the training data (and also separate from any testing data). When computational cost is prohibitive, *greedy search* is also applied. A different search strategy is *random search*, where random choices of hyperparameter sets are tested up to a pre-defined computational budget, and the best performing model is chosen. This is gaining popularity due to empirical evidence that it can find similarly good models to pure grid search within a small fraction of the computation time (Bergstra and Bengio 2012). Due to the large number of hyperparameters of our models, we use random search for the experiments in this work.

Validation performance can also be used to conduct *early stopping*. This technique aims to find an optimization iteration prior to which the training procedure starts overfitting (Goodfellow, Bengio and Courville 2016). It does so by evaluating validation performance (most commonly loss, but also other performance metrics are used) every fixed number of iterations, and choosing the best model.

2.2.2 Deep learning approaches to survival analysis

Most deep learning approaches to survival analysis in the literature have focused on extending Cox regression, beginning with Faraggi and Simon's (1995) model. This

allows for non-linear relationships between the features and relative patient risk, replacing the linear function with a single-layer neural network. Further work by Katzman et al. (2016) presents the model DeepSurv, where the network is extended to a deep feedforward neural network. The model by Luck et al. (2017) follows a similar approach, using a neural network to predict a single patient risk score. A different type of neural networks was applied to the hazards model in DeepConvSurv (Zhu, Yao and J. Huang 2016), using a convolutional neural network to deal with image data. In all the above models, whilst the original linear proportional hazards assumptions are relaxed, one drawback remains: Time-varying relationships between the feature and survival distribution are not captured.

In the competing risks setting, a state-of-the-art approach that has done away with the hazards model completely is DeepHit (Lee et al. 2018). DeepHit works by discretising the time horizon, and modelling incidence probabilities directly as the output of a neural network. Empirical results in the original paper show that DeepHit outperforms the conventional methods Cox and Fine-Gray on a variety of real-world and synthetic datasets. In this work, we develop new improved deep learning survival analysis models, using DeepHit's architecture as a starting point. We describe DeepHit in detail in Section 4.1.

2.3 Dimensionality reduction

Real-world data, such as images or raw survey responses that have not been filtered, usually has a high dimensionality. However, high-dimensional spaces come with undesirable properties such as the *curse of dimensionality*: In a high-dimensional space, data becomes relatively sparse so that it becomes more difficult to determine similarities and dissimilarities between data points. In machine learning, this can result in higher probabilities of error (Trunk 1979). This is related to a common problem in machine learning, and particularly deep learning, called *overfitting*, where an algorithm with insufficient training data starts fitting sampling noise on the dataset (Srivastava et al. 2014).

In machine learning settings with high-dimensional inputs it can thus be useful to reduce the dimensionality to a lower dimensional space that represents the "essence" of the data (Murphy 2012). Such techniques are called *dimensionality reduction*, and often result in better prediction accuracy. By reducing the volume of data we may also reduce the computational burden, resulting in shorter training and prediction times.

Approaches to dimensionality reduction can be classified into *feature projection* and *feature selection*. In feature projection, the original data is mapped into a lower-dimensional subspace. The most common approach is *principal components analysis* (PCA) (Murphy 2012), which uses a linear transformation whilst still capturing the largest possible variance from the original representation. More novel methods attempt dimensionality reduction in a non-linear way, such as *auto-encoders* (Murphy 2012) that use neural networks trained to predict the input itself. A small *bottleneck* layer in the auto-encoder then yields the desired representation.

Feature projection has the disadvantage that the original semantics of features are not preserved. Medical applications however often require the interpretability of machine learning methods or their inputs to achieve acceptance by practitioners and withstand legal scrutiny (Ching et al. 2018; Saeys, Inza and Larrañaga 2007). In this work we therefore focus on *feature selection*, in which traditionally a subset of the original features is returned, offering the advantage of interpretability by a domain expert.

2.3.1 Feature selection

In the strictest sense, feature selection methods operate by choosing a subset of all original features, aiming to remove those features that are not relevant to the prediction task. We denote this as *hard* feature selection in order to distinguish *soft* model-specific methods that attempt to reduce focus on certain features without eliminating them entirely.

Here we present the three main classes of hard feature selection techniques: *Filter*, *wrapper* and *embedded* methods. We follow this with a discussion of both hard and soft feature selection approaches for deep learning models in Section 2.3.2.

2.3.1.1 Filter methods

Filter methods are the simplest and often most convenient form of feature selection and are independent of the machine learning model. However, they thus also have the drawback of not being able to include potential model interactions when selecting features (Saeys, Inza and Larrañaga 2007).

Most filter methods are *univariate*, that is they rank features individually based on a relevance measure and select the top K, where K is a choice on the tradeoff between accuracy and complexity (Murphy 2012). By addressing features individually, these methods ignore potential relationships between variables. However, they

have been shown to be computationally efficient and robust to overfitting (Hamon 2013). Other algorithms are *multivariate*, attempting to combat the drawbacks of univariate techniques by considering feature dependencies. Methods also differ on whether features and target variables are assumed categorical or continuous. In the following we present the most common techniques for categorical target variables (i.e. classification tasks), corresponding to predicting a finite number of events in the context of survival analysis.

ANOVA and t-test Analysis of variance (ANOVA) is used to analyse differences among group means in a sample. The corresponding hypothesis test investigates whether it is statistically significant that these means are different between groups under the assumption of a Gaussian distribution. When used for feature selection, the data sample is split into groups by the class label, and the group means for a given feature are compared. The significance value for whether the sub-sample means are different is obtained by considering the F-statistic that captures the ratio of *between-group variability* to *within-group variability*, and comparing it to the corresponding F-distribution (DeGroot and Schervish 2013). In this dissertation we use ANOVA in a setting where the class label is binary. In this case, the F-test is equivalent to performing Student's t-test for difference of two sample means. Given two samples X_1 and X_2 of size n_1 and n_2, the t-statistic is given by

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{(n_1-1)s^2_{X_1}+(n_2-1)s^2_{X_2}}{n_1+n_2-2}}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}, \tag{2.34}$$

where $\bar{X}_i$ denotes the sample mean and $s^2_{X_i}$ the respective sample variance. To obtain significance values, the t statistic is compared to a t distribution with $n_1 + n_2 - 2$ degrees of freedom.

SVM weights A novel method for feature selection is to use the weight vector of a trained support vector machine (SVM) classifier (Bron et al. 2015). The classifier is of the form

$$y = \begin{cases} 0, & \text{if } \mathbf{w}\cdot\mathbf{x}+b \leq 0 \\ 1, & \text{if } \mathbf{w}\cdot\mathbf{x}+b > 0, \end{cases} \tag{2.35}$$

for input $\mathbf{x}$, output y and parameters $\mathbf{w}$ and b. Assuming normalized (mean 0 and variance 1) features, each component w_i of the weight vector can be thought of as a measure of how much feature i impacts the final classification score. A higher absolute weight would imply a feature is more relevant. This is a multivariate filter method, as the SVM is trained with the complete feature set.

Relief-based methods Relief-based feature selection methods are a family of multivariate filter methods based on the original algorithm by Kira and Rendell (1992). They operate by testing how well features distinguish between class instances that are near to each other: When two instances of the same class are near to each other but features are different, the feature scores decrease. However, if an instance pair has a different class, those features that differ for the two instances receive a higher score. Various extensions have been proposed, including the ReliefF algorithm (Kononenko, Šimec and Robnik-Šikonja 1997) that was designed to additionally handle multi-class and incomplete data.

Mutual information and χ2-test of independence These two methods are popular univariate relevance measures for feature selection when both the class label and features are discrete. Heuristically, mutual information is the reduction in entropy on the distribution of class labels once the value of feature j is observed (Murphy 2012), and is calculated via empirically estimated probabilities from a dataset. The χ^2 statistic tests independence of two events, and in the context of feature selection can be used to determine whether the occurrence of a particular feature is independent of the class label. To compute it, one compares the expected number of occurrences to the actual observed counts for each feature value-class label combination (DeGroot and Schervish 2013). In this dissertation we do not use these two methods, as our datasets contain continuous features.

Other filter methods Whilst it is beyond the scope of this dissertation to review all filter feature selection methods, we have presented some of the most popular methods above. There is vast further literature on this topic, with additional methods including *correlation-based feature selection* (Hall and Smith 1999), *Markov blanket filters* (Koller and Sahami 1996) and *fast correlation-based feature selection* (Yu and Liu 2003).

2.3.1.2 Wrapper methods

Wrapper methods select features by fitting several models on subsets of features, and choosing the best performing model. By construction, these take into account relationships between variables, an advantage over filter methods. However, wrapper methods require significant computation and sometimes prohibitive time as they are required to retrain the model at every iteration.

The most simple methods are *greedy*-based (Murphy 2012). In *forward selection* one begins with no features, and adds the best feature at every step. The best feature is chosen by training one model for each potential feature and comparing a validation performance metric. *Backward selection* begins with all features in the model, and deletes the worst feature at each step. More sophisticated search strategies have also been proposed, such as *beam search* (Siedelecki and Sklansky 1988) and *plus q take-away r* (Ferri et al. 1994). There is also a class of randomized algorithms, including *randomized hill climbing* (Skalak 1994) and genetic algorithms (Holland 1992).

2.3.1.3 Embedded methods

Embedded methods are a *hybrid* between filter and wrapper methods, using characteristics of the machine learning model itself to develop a feature selection. The idea is to improve on the performance of filter methods whilst being far more computationally efficient than wrapper methods (Saeys, Inza and Larrañaga 2007). Examples of embedded methods are within *random forests/decision trees* (Anaissi, Kennedy and Goyal 2011) or the *weight vector of SVM* (Ooi and Tan 2003). Lasso regularisation is another embedded technique that can lead to sparse solutions for linear models (Murphy 2012), and can also be applied for different models and loss functions. As embedded methods are model-specific, we discuss approaches that have been used for deep learning in Section 2.3.2.

2.3.2 Feature selection in deep learning

Whilst feature selection is standard for many machine learning models, deep learning's success at reducing the need for feature engineering (Goodfellow, Bengio and Courville 2016) may have reduced the focus on feature selection for neural networks. However, there is some evidence in the literature that deep learning approaches can indeed benefit from feature selection. In addition to achieving better generalization capabilities (Kavzoglu and Mather 2002), a significant benefit could also be faster run times and smaller memory requirements (Roy, Murty and Mohan 2015). But recent work in this area is relatively sparse. No generally accepted method for feature selection for deep neural networks exists, with few efforts to compare a variety of methods, with the exception of Kordos et al. (2017).

Filter methods One approach to feature selection in deep learning, often selected for its computational simplicity, is applying a filter method in advance to network training. In the literature, evidence of significant performance improvements however

is rare: Recent work either achieves only small performance gains, or does not give benchmarks versus the full model:

- Various filter methods are compared in (Kordos et al. 2017), including Information Gain, Information Gain Ratio, Deviation, Chi Squared, Gini Index, Uncertainty, SVM, Correlation, Rule and Relief. Only Relief (Kira and Rendell 1992) and SVM-based (e.g. Bron et al. 2015) filters could yield a marginal improvement in model classification accuracy over when no feature selection was used. Ruangkanokmas, Achalakul and Akkarajitsakul (2016) use χ^2 values to pre-select from over ten-thousand features (due to word embeddings), although no performance comparison is made with a model including all features.

- Attempts have also been made in combined use of multiple filter methods: Tsai and Hsiao (2010) fit three models using the output of different dimensionality reduction techniques: Decision trees and genetic algorithms (for feature selection), as well as PCA (for feature projection). Best results were achieved by combining all three methods, using the intersection of features selected by each algorithm, outperforming models where the union was used. Multiple feature selection methods are also used by Mares, Wang and Guo (2016), where algorithms for feature selection are classified into three groups by their underlying assumptions: Linear, non-linear and multivariate feature effects. One method for each category is used, and best results are achieved by using the union of features selected by each algorithm, achieving similar performance to a full model.

Wrapper methods Various wrapper methods have been proposed and tested, including extensions of standard methods tailored to neural networks. These methods all post performance gains, however at a significant and sometimes impractical computational cost:

- Forward, backward and evolutionary selection wrapper methods are tested by Kordos et al. (2017). Performance is found to be marginally improved, albeit at a computational cost two to four orders of magnitude higher than for filter methods. An *ant colony optimization* search strategy is used by Sivagaminathan and Ramakrishnan (2007), yielding significantly improved testing accuracy on medical classification tasks. A wrapper for pairwise feature search is devised by Michalak and Kwasnicka (2006), attempting to improve computational efficiency by using feature correlations to rule out tests for certain feature pairs.

- There have also been approaches to using heuristics derived from the neural network to guide the search: Setiono and Liu (1997) determine the least important feature to remove at each iteration by considering classification accuracy when each feature is set to 0. The final model with selected features was shown to outperform the original model with all features. A weight analysis-based heuristic was introduced by Hsu, H.-J. Huang and Dietrich (2002).

- Kabir, Islam and Murase (2010) devise a wrapper method that also modifies the architecture of the neural network by adding neurons in hidden layers during feature optimization.

Embedded methods Most embedded methods aimed at feature selection for neural networks use some form of analysis of the trained network's weights:

- Methods in (Kordos et al. 2017), (Roy, Murty and Mohan 2015) and (Belue and Bauer 1995) consider the first layer by summing up absolute values of the weights that are connected to each feature. These "activation potentials" are proposed as a metric for the importance of the feature in the network. An extended idea is used by Bauer, Alsing and Greene (2000), benchmarking each feature versus a synthetically generated noise feature to test for relevance. The most recent results from Kordos et al. (2017) show no improvement over using all features, whilst Roy, Murty and Mohan (2015) do find an improvement both over the full model and over dimensionality reduction to the same number of features via PCA.

- Belue and Bauer (1995) use an alternative metric including the partial derivative of network output with respect to each feature, yielding similar results.

Soft feature selection Whilst any regularisation technique can be seen as a way to reduce network weights, deep learning architectures have been proposed to specifically address feature weights in the philosophy of soft feature selection. Two approaches inspire models we develop in this work:

- An architecture that encourages soft feature selection is presented by Li, C.-Y. Chen and Wasserman (2016) and called Deep Feature Selection (DFS). DFS adds a sparse linear layer to the beginning of the neural network with one-to-one connections between the ith hidden node and the ith feature. Features are selected by introducing L_1 (lasso) and L_2 regularisation that encourages sparsity, reducing the weights for the sparse connections.

- Attention mechanisms for neural networks have recently gained popularity in the machine learning literature and may also be viewed as a way of soft feature selection. The general technique of attention is to multiply features by a mask of weights between 0 and 1 that are themselves learnt by a sub-network. The concept originally stems from sequence-to-sequence modelling tasks, where significant performance improvements have been made in applications such as speech recognition (e.g. Chorowski et al. 2015), and machine translation (e.g. Bahdanau, Cho and Bengio 2014; Vaswani et al. 2017). In these applications, attention allows the network to focus on different specific parts of the input sequence for different parts of the output. In visual attention (e.g. Jetley et al. 2018), the idea is to adaptively shift focus to a part of the image depending on the image itself and the model's task.

Chapter 3

Prediction task

We focus our analysis on a specific task of particular medical relevance: Prediction of prostate cancer incidence. To this end, we create a competing risks setting on a real-world dataset derived from the *Prostate, Lung, Colorectal and Ovarian Cancer Screening Trial* (PLCO). To the best of our knowledge, it is the first time the dataset is used for this prediction task. Whilst prostate cancer incidence prediction is our main objective, we additionally use the *Surveillance, Epidemiology, and End Results Program (SEER)* dataset to benchmark our machine learning methods. For SEER, we also perform experiments on an extended synthetic dataset (which we call *SEER-synth*) with additional 100 randomly generated features independent of the survival time and event label in order to investigate the impact of large feature sets on deep learning methods.

3.1 Medical relevance

Prostate cancer accounted for an estimated 19.5 annual deaths per 100 000 US males in 2011-15, that is 10% of all cancer deaths making it the second most deadly site after lung and bronchus cancer (Noone et al. 2017). In the UK, 13% of male cancer deaths in 2014 were due to prostate cancer (*Prostate cancer mortality statistics, Cancer Research UK* 2018).

In order to detect prostate cancer early, the two common methods of screening are the prostate specific antigen (PSA) blood test and digital rectal examination (DRE). Two randomized, controlled trials of prostate cancer screening were conducted to examine the impact of screening on prostate cancer mortality: The Prostate, Lung, Colorectal and Ovarian (PLCO) Cancer Screening Trial in the US and the European Randomized Study of Screening for Prostate Cancer (ERSPC) (Gohagan et al. 2000; Schröder et al. 2009).

The results of these trials are conflicting: Whilst the PLCO trial could not find a reduction in prostate cancer mortality in the screening (intervention) arm versus the control arm (Andriole et al. 2009, 2012; Pinsky et al. 2017), the ERSPC showed a statistically significant relative reduction of 21% in prostate cancer mortality for the screened cohort (Schröder et al. 2009, 2012, 2014). However, the ERSPC also concluded that despite the significant reduction in mortality, the time for population-based screening has not yet arrived due to the harms associated with a high rate of overdiagnosis and overtreatment that weigh against the benefits of screening (Schröder et al. 2014).

An overdiagnosed cancer may be defined as one that would not have been symptomatic or detected in the absence of screening, and thus treatment of overdiagnosed cancers would not be required (this definition is based on hindsight). The harms associated with such treatment have been shown to diminish the benefit of PSA screening in terms of quality-adjusted life-years by Heijnsdijk et al. (2012), who recommend that further analyses are done before universal recommendations regarding screening programs can be made.

The screening policy of the PLCO and ERSPC trials was to advise patients with positive screening results to seek diagnostic evaluation (through biopsy) of prostate cancer. Both used minimum PSA thresholds to indicate a positive result: 4 ng/ml in PLCO (Prorok et al. 2000), and most centers used 3 ng/ml in ERSPC (Schröder et al. 2009). Despite advantages through its simplicity, a single cutoff level however ignores the full patient measurements and characteristics that could be used to improve the predictive value of whether a cancer will become symptomatic and thus should mandate a referral to biopsy.

There have been various attempts in the literature to reduce overdiagnosis or combat its effects. Individualised estimates of overdiagnosis based on age, PSA level and Gleason score (which is determined on biopsy), as described by Gulati et al. (2014), inform patients and doctors on the risks involved and could be used to guide decisions on appropriate treatment. This approach however does not add predictive power to whether a given patient will develop clinical prostate cancer.

A different approach would be to identify low- and high-risk subgroups of the population, and altering their respective screening schedules. Vickers et al. (2013) do this based on age and a previous mid-life PSA measurement, whilst Pashayan, Duffy et al. (2015), Pashayan, Pharoah et al. (2015) and Yen et al. (2015) use a polygenic risk score to stratify the population into relevant subgroups for screening. Drawbacks here would be the availability of a polygenic risk score across the wider population,

and there is still scope for further development in the methods (Vickers et al. (2013) only use a single PSA measurement to stratify the population).

3.2 Question for study

Our aim is to improve and better understand predictions of prostate cancer incidence based on screening data, in order to facilitate approaches to combat overdiagosis such as (Pashayan, Duffy et al. 2015; Pashayan, Pharoah et al. 2015; Yen et al. 2015). Our individualized risk predictions may also be used for referral decisions to biopsy (analogous to Thompson et al. 2006).

We base our risk predictions on baseline prostate cancer screening data that could be readily available for a large part of the population. We also introduce a competing risks setting, with *death* as an additional terminating event. This improves the quality of risk scores obtained, since no screening would be needed for a patient who will not develop prostate cancer in his lifetime.

3.3 Description of PLCO dataset

The Prostate, Lung, Colorectal and Ovarian (PLCO) Cancer Screening Trial provides a comprehensive dataset to analyse its prostate cancer screening, incidence and mortality results. Its design has been previously described by Prorok et al. (2000). Between 1993 and 2001, participants aged between 55 and 74 were enrolled in 10 US study centers and randomised into a screening and control group. Those in the screening group were offered annual PSA testing for 6 years and annual digital rectal examination for 4 years, whilst those in the control group received normal care. For both groups, all diagnosed cancers and deaths were collected and monitored, whilst screening results were recorded only for the screening group. In this analysis we restrict our dataset to the screening group in order to target predictions based on screening data.

The event of interest is prostate cancer incidence (Event 1). Furthermore, the participant could have been right-censored prior to prostate cancer diagnosis, or have died, which is included as a separate event (Event 2). The number of participants for each event are as follows:

- **Total patients:** 38,052
  - **Event 1:** Prostate cancer incidence: 4,425 (12%)

- **Event 2:** Death prior to prostate cancer incidence: 6,032 (16%)
- **Censored:** 27,595 (73%)

There are 66 original features in the dataset. After one-hot encoding of categorical variables we obtained 105 feature columns used in the models:

- Trial entry / year 0 PSA level (1 feature) and DRE result (1 feature - 6 features after one-hot encoding)
- PLCO background questionnaire (64 features - 99 features after one-hot encoding) (demographics, smoking, (prostate) cancer family history, body type, NSAIDS, diseases, male specifics, prostate surgery)

The dataset had 13% missing values (prior to one-hot encoding), however all but 15 features had missingness of 5% or below. PSA level at year 0 had missingness of 10%. The remaining features with high missingness were mostly questions that were left out for large parts of the cohort, such as details around smoking and medical conditions if the patient had indicated that these conditions apply to him. Missing values were imputed using the mode for binary/categorical features, and the mean for numerical features. All features were furthermore normalised to mean 0 and variance 1 prior to training the models.

For survival analysis, we set time 0 to be the time of the first PSA or DRE screen, whichever occurred later. The average exit months for the trial were 122 months, with the distribution by event (including median and interquartile ranges) shown in the box plots of Figure 3.1.

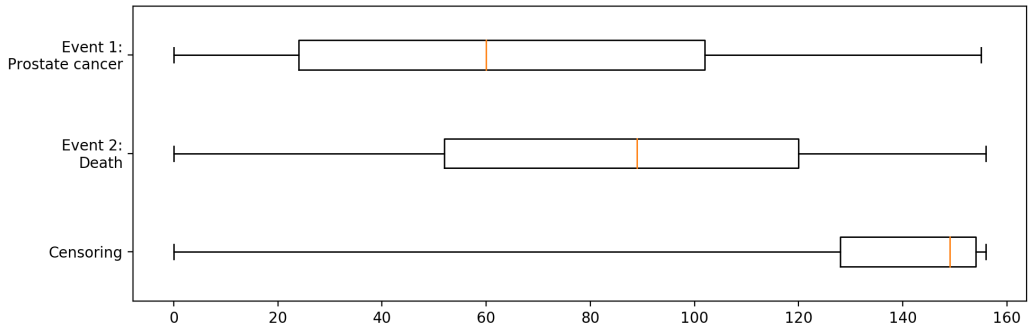


Figure 3.1: PLCO: Distribution of survival times for each event (in months)

3.4 Description of SEER dataset

The Surveillance, Epidemiology, and End Results Program (SEER)[1] is a publicly available dataset. We use an extracted cohort of 72,809 breast cancer patients from 1992 to 2007. The event of interest is death, which occured either from breast cancer, cardiovascular disease or other causes. Patients living at the end of the study were right-censored. The number of patients by event are as follows:

- **Total patients:** 72,809
  - **Event 1:** Death from breast cancer: 10,634 (15%)
  - **Event 2:** Death from cardiovascular disease (CVD): 903 (1%)
  - **Event 3:** Death from other causes: 4,484 (6%)
  - **Censored:** 56,788 (78%)

SEER: Original features The SEER dataset contains 23 features:

- Demographics, including age, race, gender (8 features)
- Morphology information (lymphoma subtype, histological type, etc.), diagnostic information, therapy information, tumor size and tumor type (15 features)

Missing values were imputed using the mean value for real-valued features and the mode for categorical features. All features were furthermore normalised to mean 0 and variance 1 prior to training the models.

SEERsynth: Additional synthetic features In order to compare the performance of DeepHit when there are a large number of irrelevant features, we also artificially extended the SEER dataset with 100 synthetic features. We distinguish this dataset from the original *SEER* dataset by denoting it as *SEERsynth*.

The 100 additional features are constructed for each patient independently from both other features and the survival distribution (time and event). For each feature j we initially pick a parameter p_j, where

$$p_j \sim \text{Unif}[0, 1], j = 1, \ldots, 100. \tag{3.1}$$

Finally, feature values for each patient are drawn from the distribution Bernoulli(p_i). They therefore represent additional binary random noise irrelevant to the prediction

[1] https://seer.cancer.gov/causespecific/

problem. We choose binary features due to their common occurence in medical datasets either as indicators of medical conditions, or one-hot encodings of categorical features.

The SEERsynth dataset thus includes 123 features:

- Original SEER features (23 features)
- Synthetic features (100 features)

The average exit months for the SEER and SEERsynth datasets were 127 months, with the distribution by event (including median and interquartile ranges) shown in the box plots of Figure 3.2.

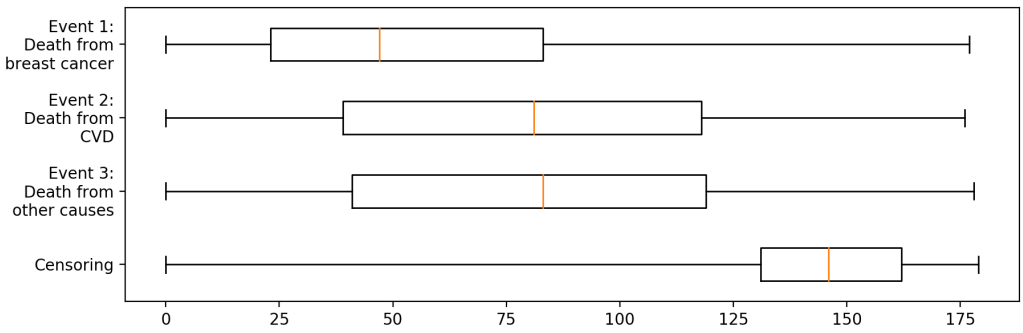


Figure 3.2: SEER: Distribution of survival times for each event (in months)

Chapter 4

Methodology

In this chapter, we first present the structure of our baseline deep learning survival model, DeepHit, including improvements we have implemented to create the model DeepHit$^+$. We continue by describing motivations for and details of several novel models using automatic feature selection techniques intended to improve performance of the survival analysis task.

4.1 DeepHit and DeepHit$^+$

DeepHit (Lee et al. 2018) is a deep learning method for survival analysis with competing risks. In contrast to many current methods for survival analysis, DeepHit does not assume a specific parametric form of the survival distribution. Furthermore, its deep learning architecture allows for very general relationships between covariates and the survival distribution. In its original paper, it was shown that DeepHit outperforms conventional methods, including Cox and Fine-Gray, on a variety of real-world and synthetic datasets.

4.1.1 Cumulative incidence function estimation

DeepHit works by treating survival times as discrete, with time set $\mathcal{T} = \{0, \ldots, T_{\max}\}$, for example representing monthly intervals. Given input features $\mathbf{x}$ it produces an output vector

$$\mathbf{y} = [y_{1,1}, \ldots, y_{1,T_{\max}}, \ldots, y_{K,1}, \ldots, y_{K,T_{\max}}]. \tag{4.1}$$

An output component $y_{k,s}$ corresponds to the estimated probability $\hat{P}(s, k \mid \mathbf{x})$ that the patient will experience event k at time s. The estimated cause-specific cumulative

incidence function thus becomes

$$\begin{aligned}\hat{F}_k(t \mid \mathbf{x}) &= \hat{P}(s \leq t, k \mid \mathbf{x}) \\ &= \sum_{s=0}^{t} \hat{P}(s, k \mid \mathbf{x}) \\ &= \sum_{s=0}^{t} y_{k,s}.\end{aligned} \tag{4.2}$$

4.1.2 Architecture

DeepHit's architecture is a multitask network (Collobert and Weston 2008) that comprises a shared sub-network, and K cause-specific sub-networks shown in Figure 4.1 (adapted from Lee et al. 2018). The cause-specific networks receive the output $f_s(\mathbf{x})$ of the shared network as well as a *residual connection* of the original features $\mathbf{x}$ as input. This ensures that the sub-networks can learn from both the common representation and individual cause-specific features.

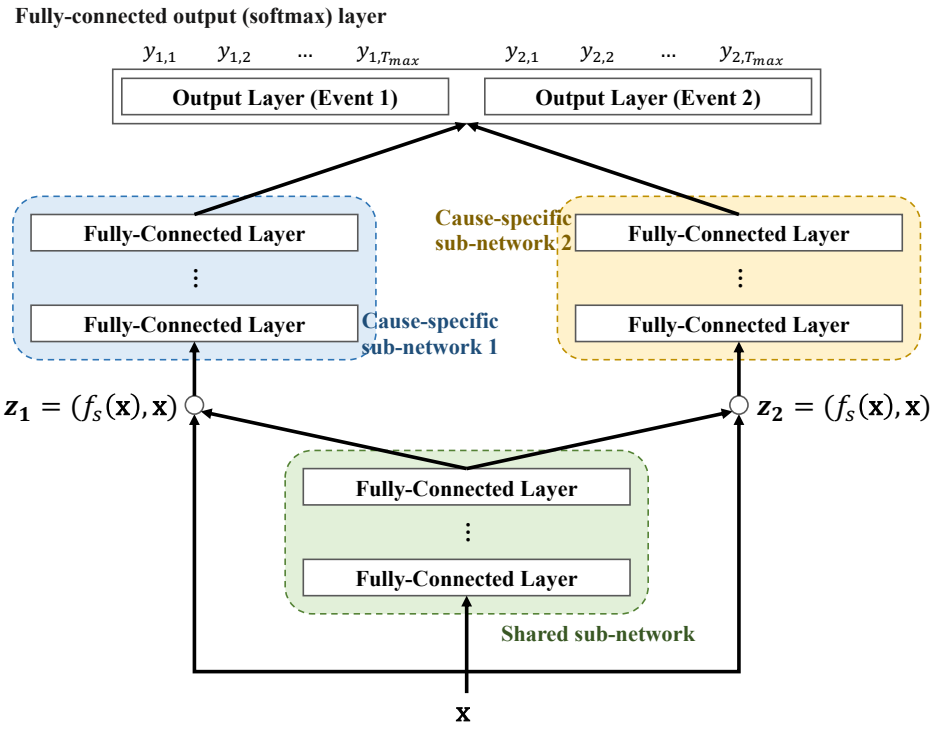


Figure 4.1: DeepHit architecture with two competing risks

The output layer $\mathbf{y}$ becomes a probability distribution by using a softmax activation function. It is also fully connected with all outputs of the cause-specific sub-networks.

4.1.3 Loss function

The loss function for DeepHit is specifically designed to handle censored data, and is the sum of two terms $\mathcal{L}_{\text{Total}} = \mathcal{L}_1 + \mathcal{L}_2$.

$\mathcal{L}_1$ is the negative log-likelihood of the joint survival distribution for all competing risks, adapted for right-censoring. It follows the same structure as in Section 2.1.3, and is given by

$$\mathcal{L}_1 = -\sum_{i=1}^{N} \left[\mathbf{1}_{\{k^{(i)} \neq \emptyset\}} \log(y^{(i)}_{k^{(i)},s^{(i)}}) + \mathbf{1}_{\{k^{(i)}=\emptyset\}} \log\left(1 - \sum_{k=1}^{K} \hat{F}_k\left(s^{(i)} \mid \mathbf{x}^{(i)}\right)\right)\right]. \tag{4.3}$$

In addition to $\mathcal{L}_1$, which drives the network to learn the general joint distribution, DeepHit incorporates a second term $\mathcal{L}_2$ intended to fine-tune the network to discriminate estimated individual risks for each cause. The idea is an adaptation of concordance (Harrell et al. 1982): An individual who experiences an event at time s should have higher risk at time s than a patient who survived longer than s. Patient pairs (i, j) where we can compare such risk for event k are denoted by the indicator function $A_{k,i,j} = \mathbf{1}_{\{k^{(i)}=k, s^{(i)}<s^{(j)}\}}$. The second loss term is then defined as

$$\mathcal{L}_2 = \sum_{k=1}^{K} \beta_k \sum_{i \neq j} A_{k,i,j} \cdot \eta\left[\hat{F}_k\left(s^{(i)} \mid \mathbf{x}^{(i)}\right), \hat{F}_k\left(s^{(i)} \mid \mathbf{x}^{(j)}\right)\right]. \tag{4.4}$$

The authors set $\beta_k = \beta$ for all k, and $\eta(x, y) = \exp(-\frac{x-y}{\sigma})$ for hyperparameters β and σ to be decided.

4.1.4 Model settings and hyperparameters

In the original implementation by Lee et al. (2018), all hidden nodes used ReLU activation functions. The shared sub-network consisted of 1 fully-connected layer, whilst the cause-specific sub-networks were set to 2 fully connected layers (in addition to the output layer). The number of nodes per hidden layer was chosen to be 3, 5, and 3 times the feature dimension for layers 1, 2, and 3, respectively. Loss function hyperparameters β and σ were chosen by random search based on discriminative performance on a validation set. DeepHit was trained using the Adam optimizer with learning rate of 10^{-4}, minibatch size of 50, Xavier initialization and dropout probability of 0.6. Furthermore, early stopping was performed using the validation loss.

4.1.5 DeepHit$^+$ improvements

DeepHit$^+$ implements two improvements on DeepHit's published model with adaptations to early stopping and sub-network sizes. As outlined in Chapter 5, these improvements on their own trigger significant performance gains over the original model. DeepHit$^+$ is further used as base model for the automatic feature selection methods that we develop in this work.

Early stopping In the original implementation, DeepHit was trained using an early stopping procedure using validation loss. In DeepHit$^+$ we change the metric from loss to the performance measure C-index (see Section 5.1) in order to make the early stopping procedure more accurate. At the expense of some computational time, this aligns more closely with the ultimate prediction objective, as C-index is used to evaluate model performance on the test dataset.

Network sizing Whilst the number of layers were fixed, DeepHit had only semi-adaptive layer sizes, with the hidden dimension (number of hidden nodes) being a fixed multiple of the input size. Several reasons however motivate us to deviate from this fixed sizing:

- As outlined by Goodfellow, Bengio and Courville (2016), the size of a neural network should be chosen based on a trade-off between desired complexity for good generalization and required simplicity for ease of training. This trade-off is task-specific, and may not solely depend on the number of input features.
- In multitask learning settings, the above trade-off applies to each of the network's tasks, motivating the need for differing sub-network sizes.
- Both the number of layers and the number of hidden nodes in each layer are parameters of an optimal architecture size.

In order to improve the model's flexibility and resilience against overfitting, we therefore modify the network architecture to allow arbitrary numbers of layers, as well as sizes of each layer. In particular, we allow the shared sub-network and each cause-specific sub-network to be differently sized. For training DeepHit$^+$ we then expand random search across hyperparameters to additionally include the

1. number of layers in the shared sub-network, n_s,
2. hidden dimension of layers in the shared sub-network, h_s,

3. number of layers in each cause-specific sub-network, $n_k, k = 1, \ldots, K$, and the
4. hidden dimension of layers in each cause-specific sub-network, $h_k, k = 1, \ldots, K$.

4.2 Considerations for automatic feature selection

In the following sections of this chapter we propose various models with automatic feature selection for survival analysis with deep learning. We use DeepHit$^+$'s architecture as a basis for our methods. A special consideration when using deep learning is its computational complexity, as both backpropagation and repeated C-index calculations are computationally expensive. For this reason, wrapper feature selection methods (c.f. Section 2.3.2) are not feasible for large modern datasets and deep learning architectures, as they require to train the network multiple times (often at least linear in the number of features).

We therefore focus on filter and embedded approaches to hard feature selection, as well as soft feature selection techniques for deep learning:

- In FilterDeepHit$^+$ we construct several applications of filter feature selection methods to deep survival analysis. These use the simplest form of feature selection as a pre-processing step independent of the neural network used for survival analysis.
- We build embedded approaches in HybridDeepHit$^+$: These use the neural network itself to create feature importance scores to conduct feature selection, and the neural network is subsequently retrained on an optimal subset of features.
- Our two soft feature selection approaches build special architectures in the neural network itself to aid self-guided feature selection. In SparseDeepHit$^+$ this is a sparse regularisation layer, whilst in AttentiveDeepHit$^+$ we use a separate attention network.

4.3 FilterDeepHit$^+$: Automatic feature selection using filter method

4.3.1 Motivation

Filter feature selection methods are very common and often computationally cheap, and thus have the advantage of wide applicability. On the PLCO dataset, their application indicates that only a small subset of features are relevant for predicting the

survival distribution. For example, Figure 4.2 shows the p-values of all 105 features from an ANOVA F-test (or equivalently Student's t-test) using Event 1 incidence at 120 months. Of all features, fewer than 50 have a p-value less than 0.1, motivating that more than half of the features could be removed.

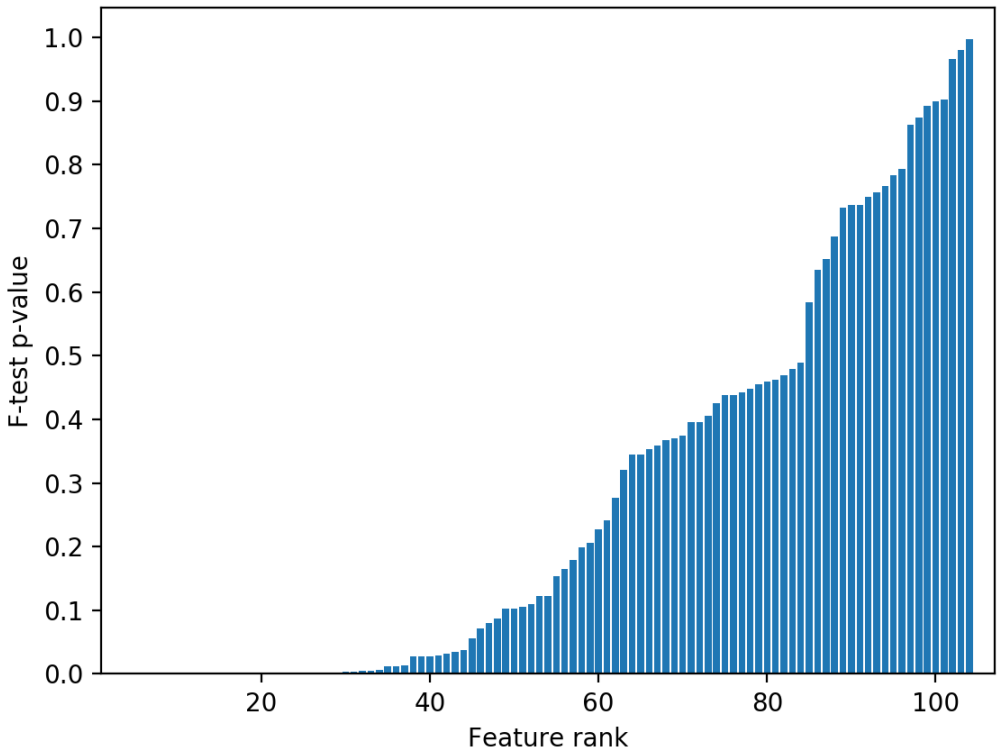


Figure 4.2: ANOVA F-test p-values of PLCO features for Event 1 incidence at 120 months. A lower p-value indicates higher feature relevance.

As is evident from Figure 4.2, filter methods can give valuable information to reduce a model's input features. Based on this premise, we develop a family of models we call FilterDeepHit$^+$. This requires special care in the adaptation of filter methods to work for survival analysis, as well as modifications to the deep learning architecture.

4.3.2 Modifications of DeepHit's architecture

Whilst it would be possible to feed a subset of features in place of $\mathbf{x}$ into DeepHit's original network from Figure 4.1, we develop a technique more tailored to the setting of multitask learning and competing risks. In particular, we desire the ability to make

different feature selections for each competing risk. In order to feed selected features into each cause-specific sub-network in DeepHit$^+$, we modify its architecture. The modified architecture is shown in Figure 4.3. In the original architecture, both the shared and cause-specific sub-networks had input connections of all available features, $\mathbf{x}$. We replace these connections with certain selected subsets of the features, which we denote $\mathbf{v}_s(\mathbf{x})$ for the shared sub-network and $\mathbf{v}_k(\mathbf{x}), k = 1, 2, ...$ for the kth cause-specific sub-network.

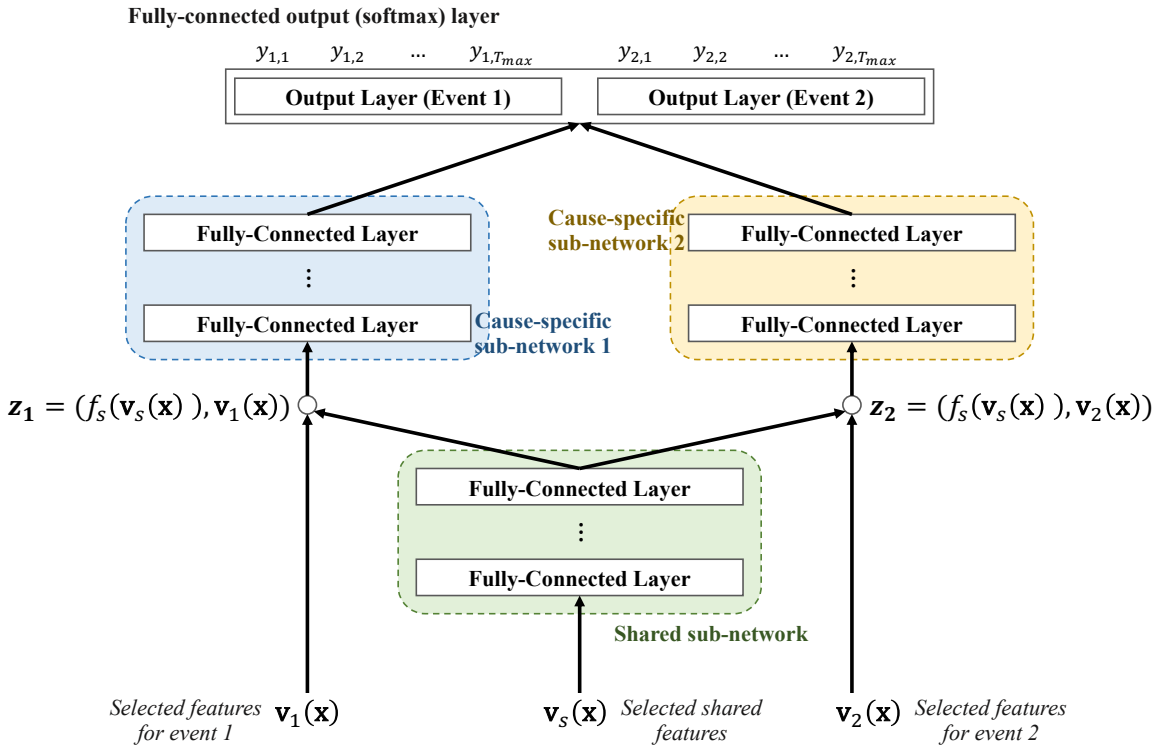


Figure 4.3: Modified DeepHit$^+$ architecture with two competing risks

In this way, the cause-specific sub-networks are fed only the pre-selected relevant features for the respective events (in addition to the shared representation). We also allow for a pre-selection of features to be fed to the shared sub-network.

4.3.3 Adaptation of filter methods to survival analysis with competing risks

Classical filter feature selection methods are developed for classification or regression tasks, and are therefore not directly applicable to survival analysis. A direct ap-

plication of filter methods with the event label, for example, would ignore the time dimension of the survival data, potentially giving a confused picture when the dataset contains a particularly heterogeneous distribution of survival times. Using only time data to rank features would lose even more information about whether a patient experienced the event or was censored.

In order to use filter methods for survival analysis, we develop a modification based on a simple observation: When fixing the time horizon to Δt, survival analysis becomes a classification task for which we can use filter techniques. In particular, if for patient i time $\tau^{(i)}$ and label $k^{(i)}$ are observed, the time-fixed label $\delta^{(i,\Delta t)}$ would be

$$\delta^{(i,\Delta t)} = \begin{cases} k^{(i)}, & \text{if } \tau^{(i)} < \Delta t \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

To account for the competing risks setting, we can further consider only incidence of a certain event k, giving the binary cause-specific time-fixed label

$$\delta^{(i,\Delta t,k)} = \begin{cases} 1, & \text{if } \tau^{(i)} < \Delta t \text{ and } k^{(i)} = k \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

For each event, we apply the filter method multiple times to score associations between features and the cause-specific label for a preselected set of evaluation time horizons. We choose these time horizons to be the same as those used for model performance evaluation (c.f. Section 5.2.1). The final score for each feature is then the average of its scores across the time horizons, and is used for automatic feature selection for the survival model.

4.3.4 Automatic feature selection

For a given train-test data split, the training data (or a subset if calculations are too expensive) is used to compute a feature scoring with the selected filter method and the procedure outlined in the previous section. Whilst this outputs a feature ranking, additional decisions must be made to determine which features are selected for input into the survival analysis model. Our method in FilterDeepHit$^+$ automatically selects the features for each cause-specific sub-network $\mathbf{v}_k(\mathbf{x})$ from the feature ranking. It chooses the top m_k features according to the ranking, treating the number of features to be selected $(m_k, k = 1, \ldots, K)$ as an additional hyperparameter. This is optimized through random search on the validation dataset together with the other model hyperparameters.

The shared sub-network's input features $\mathbf{v}_s$ are the common features that occur in all $\mathbf{v}_k$ (i.e. their intersection). If there are none, the modified version of DeepHit does not have a shared sub-network.

Given the datasets contain both continuous and categorical features, we only choose filter methods that are able to deal with both types of data. Whilst there is extensive general literature around filter feature selection methods, investigations of applications in deep learning remain sparse and most have only achieved marginal performance improvements, if any (c.f. 2.3.2). To illustrate the capabilities of filter feature selection we choose three methods. The first two, ReliefF and SVM weights, are the best performing methods in the neural network application reviewed by Kordos et al. (2017). As a further benchmark, we also apply feature selection using ANOVA/t-tests, due to their common use in practice.

We denote the three models as follows:

- FilterDeepHit$^+$ (ANOVA): Feature selection with ANOVA / t-tests
- FilterDeepHit$^+$ (SVM): Feature selection using the weights of a trained SVM
- FilterDeepHit$^+$ (ReliefF): Feature selection with ReliefF

4.4 HybridDeepHit$^+$: Automatic feature selection using feature importance

4.4.1 Motivation

Whilst filter methods can extract feature relevance values purely from the data, they ignore the interaction with the machine learning model employed (Saeys, Inza and Larrañaga 2007). Furthermore, filter methods act on the training data itself, with no out-of-sample validation, which could lead to overfitting. Some machine learning models contain natural information about the *importance* of features for their prediction that can be used for feature selection. For example, a simple linear regression model's coefficients and their p-values indicate how much and how significant each feature's contribution to the prediction is. However, deep learning models such as DeepHit$^+$ are *black box* and do not inherently yield insights into the relevance of specific features. Yet there are techniques to extract feature importance values for pre-trained black-box models, motivating us to devise a deep learning model with hybrid feature selection. The general approach is to fit a full model on all features

first, subsequently calculate feature importance values based on a validation dataset, and finally train a second model with only the most important features.

4.4.2 Feature importance

We analyse which features are important for DeepHit$^+$'s model predictions to generate scores for the feature selection process. Our calculations are based on an adaptation of Breiman and Cutler's permutation importance (Breiman 2001), which is the most frequently applied variable importance measure for random forests (X. Chen and Ishwaran 2012), another black-box model. The general idea is to track how much worse model predictions become when obscuring a feature: The lower the performance, the more important the feature is for it's original predictions. In the following, we develop an adaptation of this method to consistently work for the survival analysis of DeepHit$^+$. A schematic of the importance calculation method is shown in Figure 4.4.

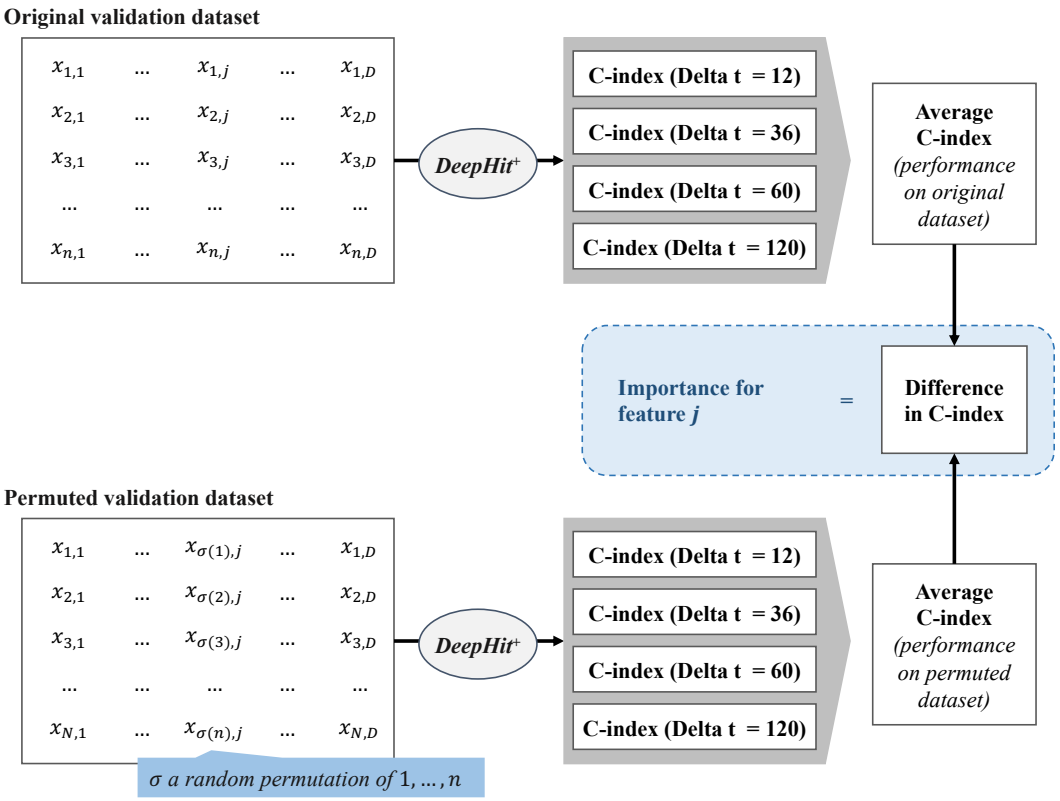


Figure 4.4: Method of calculating feature importance for a given trained DeepHit$^+$ model

Raw feature permutation importance The procedure works as follows: We initially train DeepHit$^+$ using all features. For each feature we then randomly permute its validation set values, and run the pre-trained model to predict using the new permuted validation set. The importance of the feature is defined to be the difference in true model performance (using original data) and new model performance (using the new data with the permuted feature column). Our performance metric is the C-index, which is a common measure of discriminative performance in survival analysis, and is specific to a certain event and evaluation time (for more details see Section 5.1). Feature importances are therefore event-specific, depending on the event chosen for the performance calculation. We compute the importance values for all evaluation times, and report the average.

If the C-index from the permuted data is close to the original C-index, "obscuring" the feature in this way has limited impact on the model performance, meaning the feature has low importance. If however the difference (drop) in C-index is large, the feature is more critical for the predictive value of the model.

In contrast to filter methods, where features are scored using the training dataset, we take care to calculate feature importances on the validation dataset. This is because the permutation importance method uses a model that has already been trained on the training dataset, and calculating importance on the same dataset could lead to learning overfitted features. Using the validation dataset avoids this problem so that only generalisable features score highly.

We may run the above procedure I times to obtain I different feature importance values, denoted $\lambda_{j,k,i}, i = 1, \ldots, I$ for feature j and event k. This yields a more robust mean estimate of the feature importance $\bar{\lambda}_{j,k} = \frac{1}{I} \sum_i \lambda_{j,k,i}$.

P-values Randomness in the permutations used for calculating feature importance allows testing for the true mean feature importance $\lambda^*_{j,k}$. Assuming the null hypothesis that $H_0 : \lambda^*_{j,k} = 0$, we can test for positive feature importance with the alternative hypothesis $H_1 : \lambda^*_{j,k} > 0$. Under the assumption of a Gaussian distribution we apply a one-tailed one-sample t-test on the I feature importance values. This gives the p-value for feature j on event k as

$$p_{j,k} = \mathbb{P}\left[T > \frac{\bar{\lambda}_{j,k}}{s_{j,k}/\sqrt{I}}\right], \tag{4.7}$$

where T is a random variable with Student's t-distribution on $I-1$ degrees of freedom and $s_{j,k}$ is the sample standard deviation of the raw importance values $\lambda_{j,k,i}$.

4.4.3 Automatic feature selection

For a given train-test data split, we use the validation data to calculate feature importance as described previously. There are two strategies for then automatically selecting features for the cause-specific sub-networks $\mathbf{v}_k(\mathbf{x})$ from the variable importance results:

1. We may, as in FilterDeepHit$^+$, choose the top m_k features according to their importance results, treating the number of features to be selected ($m_k, k = 1, \ldots, K$) as additional hyperparameters that are optimised with random search during training. This yields the following two models:
   - HybridDeepHit$^+$ (M) *(Using mean importance values)*: Retain only the top m_k features in $\mathbf{v}_k$ by their event-specific mean importance value $\bar{\lambda}_{j,k}$.
   - HybridDeepHit$^+$ (P) *(Using p-values)*: Retain only the top m_k features in $\mathbf{v}_k$ by their event-specific p-value $p_{j,k}$ (from the test for positive importance)
2. Alternatively, we can set importance cutoff values for each event, denoted $c_k, k = 1, \ldots, K$, that are also treated as additional hyperparameters. We obtain the two models:
   - HybridDeepHit$^+$ (M/Cut) *(Using mean importance values)*: Retain only those features in $\mathbf{v}_k$ that have an event-specific mean importance value $\bar{\lambda}_{j,k} \geq c_k$.
   - HybridDeepHit$^+$ (P/Cut) *(Using p-values)*: Retain only those features in $\mathbf{v}_k$ that when testing for positive importance have an event-specific p-value $p_{j,k} \leq c_k$.

As before in FilterDeepHit$^+$, the set of input features for the shared sub-network is the intersection of the cause-specific feature sets. HybridDeepHit$^+$ also uses the same architecture modifications to feed specific features to each cause-specific sub-network, as outlined in Section 4.3.2.

4.5 SparseDeepHit$^+$: Automatic feature selection using sparse initial layer

4.5.1 Motivation

Inspired by linear models that achieve sparsity through introducing L_1 regularisation, Li, C.-Y. Chen and Wasserman (2016) propose adding an additional sparse layer with

one-to-one connections from the input before the first hidden layer of a deep neural network. Applying regularisation to the weights of this new layer, lower weights and soft feature selection is encouraged at the input level for the neural network too. We propose a new model SparseDeepHit$^+$ by applying this concept to the multitask learning setting of DeepHit$^+$.

4.5.2 Architecture

Our architecture for SparseDeepHit$^+$ incorporates additional sparse layers for the shared sub-network as well as each of the cause-specific sub-networks. In Figure 4.5, we show this by illustrating the nodes, and connections between nodes in each layer. It is worth noting that whilst the dimension of the sparse layer is always the same as the dimension of the respective inputs, the number of nodes in each sub-network's hidden layers may be different, as in all other variants of DeepHit$^+$ depending on the choice of hyperparameters.

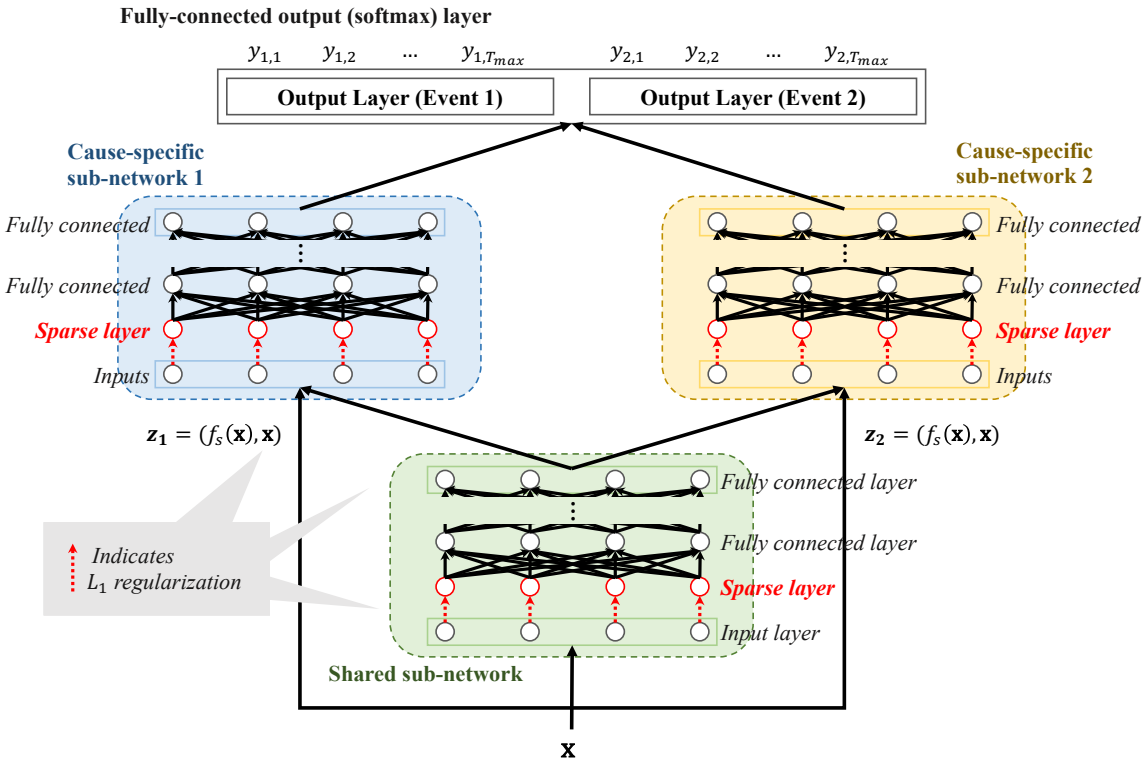


Figure 4.5: SparseDeepHit$^+$ architecture with two competing risks

Mathematically, we denote the input to the shared network $\mathbf{x}$, and the inputs

to the kth cause-specific network $\mathbf{z}_k$. Then we define parameters $\mathbf{w}_s$ and $\mathbf{w}_k, k = 1, \ldots, K$, and the activations of the sparse layers are given by

$$\mathbf{z}_s^{'} = \mathbf{x} \odot \mathbf{w}_s \tag{4.8}$$

$$\mathbf{z}_k^{'} = \mathbf{z}_k \odot \mathbf{w}_k, \quad k = 1, \ldots, K, \tag{4.9}$$

where $\odot$ denotes the element-wise (*Hadamard*) product of two vectors. These outputs $\mathbf{z}_s^{'}$ and $\mathbf{z}_k^{'}$ are then the inputs to the fully connected parts of the sub-networks as before.

The loss function, originally $\mathcal{L}_{\text{Total}} = \mathcal{L}_1 + \mathcal{L}_2$ as given in Equations 4.3 and 4.4, is adjusted to include additional L_1 regularisation terms as follows:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_1 + \mathcal{L}_2 + \gamma_s \left\|\mathbf{w}_s\right\|_1 + \sum_{k=1}^{K} \gamma_k \left\|\mathbf{w}_k\right\|_1, \tag{4.10}$$

for additional hyperparameters γ_s and $\gamma_k, k = 1, \ldots, K$.

4.6 AttentiveDeepHit$^+$: DeepHit with feature attention

4.6.1 Motivation

The promising results in visual attention for classification tasks (e.g. Jetley et al. 2018) motivate that survival analysis may also benefit from adaptive focusing on specific features. We thus develop a model applying attention to survival analysis, which we call AttentiveDeepHit$^+$. The attention mask is dependent on the input features themselves, allowing the network to detect patient-specific risk factors and feed these on for detailed survival analysis by the remainder of the network. The network's focus can be interpreted as a form of soft feature selection, as we outline below.

4.6.2 Architecture

Analogous to feature selection, we can distinguish between *hard* and *soft attention*, the former being characterized by constraining attention masks to be exactly zero or one. However, this leads to non-differentiable gradients, and thus the model cannot be trained by standard backpropagation techniques. Our model architecture therefore includes soft cause-specific attention mechanisms for the input of each cause-specific

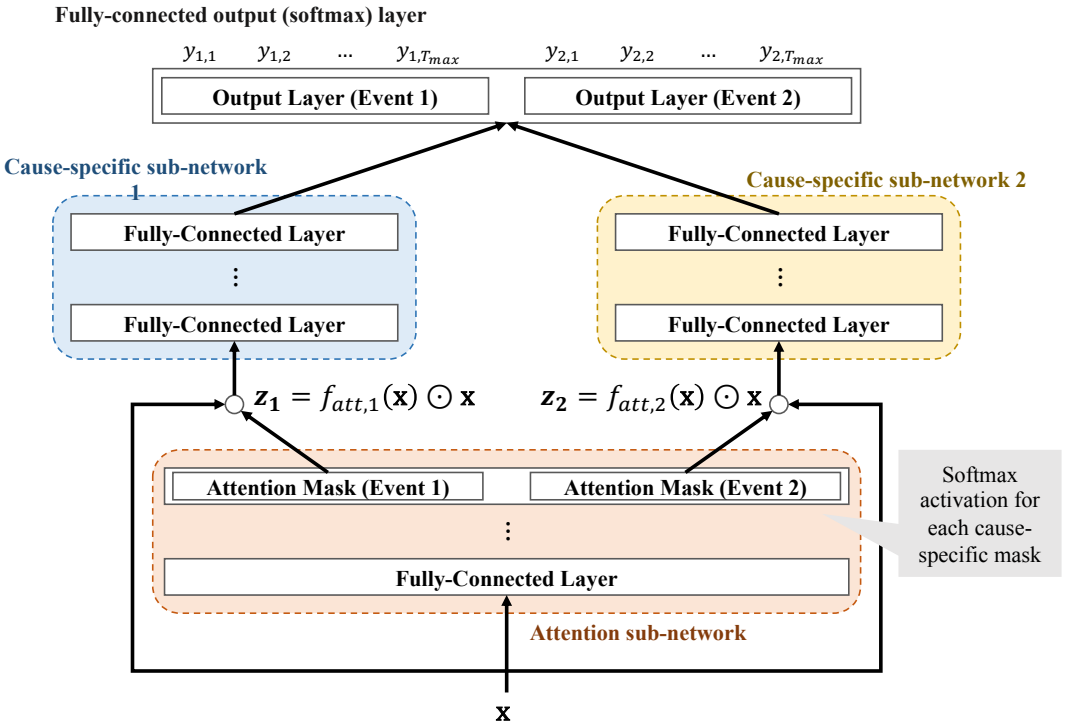


Figure 4.6: AttentiveDeepHit$^+$ architecture with two competing risks

sub-network. The architecture guarantees interpretability of the attention masks by omitting the shared sub-network. We present the overall architecture in Figure 4.6.

The attention sub-network receives all features $\mathbf{x}$ as input. It consists of several fully connected layers, and outputs one attention mask for each cause-specific sub-network, which we shall denote $f_{\text{att},k}, k = 1, \ldots, K$. The dimension of the attention mask is the same as the input vector. Furthermore, we use a softmax activation function for each of the attention masks to force the network to focus on specific features (as opposed to placing full attention on all features), since the attention masks must then sum to 1: $\sum_j (f_{\text{att},k})_j = 1$.

The inputs to the kth cause-specific sub-network $\mathbf{z}_k$ are thus given by

$$\mathbf{z}_k = f_{\text{att},k} \odot \mathbf{x}. \tag{4.11}$$

As before, the cause-specific sub-networks then comprise several fully connected layers to output the predicted survival distribution.

4.6.3 Regularisation

Whilst feature weights in the first layer of AttentiveDeepHit$^+$ are reduced by $f_{\text{att},k} \in [0,1]^D$, subsequent layers could increase weights to compensate. For this reason, we introduce regularisation for the parameters of the cause-specific sub-networks, choosing L_2 regularisation for its ease of training via gradient descent. With regularisation, the weights in cause-specific sub-networks cannot compensate for a low attention value without incurring a loss penalty.

We denote the parameters of each cause-specific network by tensors $\mathbf{W}_k$ and $\mathbf{B}_k$, and those of the final output layer as $\mathbf{w}_k$ and $\boldsymbol{\beta}_k$. The loss function, originally $\mathcal{L}_{\text{Total}} = \mathcal{L}_1 + \mathcal{L}_2$ as given in Equations 4.3 and 4.4, is then adapted as follows:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_1 + \mathcal{L}_2 + \sum_{k=1}^{K} \gamma_k \left[\|\mathbf{W}_k\|_2 + \|\mathbf{B}_k\|_2 \right] + \gamma_o \left[\|\mathbf{w}_k\|_2 + \|\boldsymbol{\beta}_k\|_2 \right], \tag{4.12}$$

for additional hyperparameters γ_o and $\gamma_k, k = 1, \ldots, K$.

Chapter 5

Experiments

In this chapter, we give details of the experimental analysis we conducted and assess the performance of our proposed models. We begin by defining the performance metric C-index that we use to compare the discriminative performance of models. This is followed by a description of the experimental setting for each model, including technical implementation details and choices of hyperparameters. For each model, we then present performance results and discuss these in the light of our benchmarks.

5.1 Performance metrics

The ordinary concordance index (C-index) (Harrell et al. 1982) is a commonly used discriminative index for survival data, encouraging correct ordering of event times. It postulates that if Patient A survived longer than Patient B, Patient A should be assigned a lower risk by the predictive model. The metric itself gives the probability of correctly ranking the event time for two randomly chosen patients of the testing sample.

We use an extention by Gerds et al. (2013) of the original C-index in order to evaluate discriminative performance at various time horizons, and adapt it to the competing risks setting. This compares the risks of pairs of patients at a certain evaluation time Δt given that one of the patients experienced event k prior to time Δt, and the other patient survived longer. Given model estimates $\hat{F}_k$ of cumulative incidence functions, patient features $\mathbf{x}$ and survival times τ, we define the C-index for event k as

$$C_k(\Delta t) = \mathbb{P}\left[\hat{F}_k(\Delta t \mid \mathbf{x}^i) > \hat{F}_k(\Delta t \mid \mathbf{x}^j) \middle| \tau^i < \tau^j, k^i = k, \tau^i < \Delta t\right]. \tag{5.1}$$

5.2 Experimental setting

In this section we describe the implementation and choices of hyperparameters for experimental analysis of the models.

5.2.1 Global settings

Cross-validation To ensure the stability of results and comparability across methods, we present the results as averages over 5 train/test dataset splits obtained from 5-fold cross-validation. All models were tested using the same data splits. We further report the standard deviation of the 5 results in order to assess the significance of differences in performance.

Evaluation times As indicated in Section 5.1, the performance metric C-index is dependent on an evaluation time. We present results for four different evaluation times: 1, 3, 5 and 10 years - or 12, 36, 60 and 120 months, respectively. This covers both short and long-term times of interest that are available in the datasets. Tables 5.1 and 5.2 show the number of events falling before each evaluation time for the PLCO and SEER datasets, respectively.

Event	$\Delta t < 12$	$\Delta t < 36$	$\Delta t < 60$	$\Delta t < 120$	Total
Event 1	648	1497	2205	3765	4425
Event 2	274	987	1783	4518	6032
Censoring	67	317	539	3501	27595

Table 5.1: Cumulative incidence of events by evaluation time for PLCO

Event	$\Delta t < 12$	$\Delta t < 36$	$\Delta t < 60$	$\Delta t < 120$	Total
Event 1	1152	4194	6408	9665	10634
Event 2	63	210	329	690	903
Event 3	352	984	1602	3386	4484
Censoring	333	597	855	3013	56788

Table 5.2: Cumulative incidence of events by evaluation time for SEER

5.2.2 Benchmarks

We benchmark performance with the most common established traditional models for survial analysis: Cause-specific Cox models (Cox 1972) (which we denote as *csCox*), Fine-Gray models (Fine and Gray 1999) and Random Survival Forests (Ishwaran et al. 2014) (denoted *RSF*). As a benchmark for deep learning models we also provide results of the original DeepHit implementation from Lee et al. (2018).

Cox We implemented cause-specific Cox proportional hazards models using the `coxph` function in the R package `survival`[1]. However, the model does not converge when the input data contains singularities. We therefore, and in an effort to reduce overfitting, perform stepwise forward-backward feature selection with a p-value threshold of 0.1. In particular, we perform the following steps:

1. Begin with an empty feature set
2. *Forward selection:* Given a selected feature set, fit models extending the feature set by one using each of the features that are not currently included.
   (a) Compare p-values for the new features from the above models. Choose the feature with the lowest p-value and add it to the feature set, provided it is less than 0.1.
   (b) For increased computational efficiency, discard any features with p-value greater than 0.1 from further analysis.
   (c) Repeat this procedure until no new features have a sufficiently low p-value to be included in the feature set.
3. *Backward selection:* Fit the model with the built-up feature set. If there are one or more features with a p-value greater than 0.1, remove the one with the largest p-value. Iterate this step until only features with a p-value below the threshold remain.

Fine-Gray We implemented Fine-Gray models using the `crr` function in the R package `cmprsk`[2], using default values for all function parameters. Again, the model fails when there are singularities in the input data. We therefore remove linearly dependent columns from the dataset prior to training the model. In this case, due to the long computation time of the model, stepwise feature selection is not feasible.

[1] https://cran.r-project.org/web/packages/survival/
[2] https://cran.r-project.org/web/packages/cmprsk/

Random Survival Forest Random Survival Forests were implemented using the `rfsrc` function in the R package `randomForestSRC`[3]. The number of trees was chosen to be 1000, and all other parameters were set to defaults.

DeepHit We implemented DeepHit, as well as DeepHit$^+$ and all its variants, using Python's `tensorflow`[4] package. The fixed model settings and hyperparameters used were as described in Section 4.1.4, apart from early stopping, which was conducted using validation C-index performance, as opposed to validation loss. This is expected to marginally improve results (c.f. Section 4.1.5).

The remaining hyperparameters β and σ were chosen by random search. This was conducted with 20 search iterations on the first cross-validation train/test split (almost amounting to exhaustive search given the small hyperparameter space). We determined the best hyperparameters based on average C-index performance across events and evaluation times on the validation dataset. During the random search procedure, values for the hyperparameters were chosen from the sets in Table 5.3. The second to fifth cross-validations used the hyperparameters determined from the first split in order to improve computational efficiency. The size of the validation dataset for random search and early stopping was 20% of the original training data.

Hyperparameter	Set of choices
β: Weight of ranking loss	0.1, 0.3, 1, 3, 10
σ: Parameter for risk comparator η	0.1, 0.3, 1, 3, 10

Table 5.3: Random search hyperparameter choices for DeepHit

5.2.3 DeepHit$^+$

In DeepHit$^+$, we conduct additional hyperparameter random search for the number of layers and hidden nodes in each sub-network. In order to focus on these variables, we fix the hyperparameters β and σ to their optimal values determined by DeepHit's initial run on the same dataset. Given the larger hyperparameter space, we conduct 50 random search iterations on the first cross-validation, and apply the optimal parameters to all other train/test splits. The sets from which the hyperparameters were chosen are given in Table 5.4.

[3]https://cran.r-project.org/web/packages/randomForestSRC/
[4]https://www.tensorflow.org/

Hyperparameter	Set of choices
n_s, number of shared layers:	1, 2, 3
h_s, nodes per hidden shared layer:	50, 100, 200
$n_k, k = 1, \ldots, K$, number of cause-specific layers:	1, 2, 3
$h_k, k = 1, \ldots, K$, nodes per hidden cause-specific layer:	50, 100, 200

Table 5.4: Random search hyperparameter choices for DeepHit$^+$

5.2.4 Extensions of DeepHit$^+$

The models FilterDeepHit$^+$, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$ are all trained using 50 iterations of random search for hyperparameters on the first cross-validation. As before, the optimal parameters are then applied to all other train/test splits. We also fix β and σ to their optimal values determined by DeepHit's initial run on the same dataset, as was done in DeepHit$^+$.

The models all include the network sizing hyperparameters from DeepHit$^+$. However, as each extension adds additional parameters, we restricted the choices for network sizing compared to DeepHit$^+$ in an effort to keep the search space manageable within the computational budget. The sets from which these shared hyperparameters were chosen are given in Table 5.5.

Hyperparameter	Set of choices
n_s, number of shared layers:	1, 2
h_s, nodes per hidden shared layer:	50, 100
$n_k, k = 1, \ldots, K$, number of cause-specific layers:	1, 2
$h_k, k = 1, \ldots, K$, nodes per hidden cause-specific layer:	50, 100

Table 5.5: Random search hyperparameter choices for the common hyperparameters in FilterDeepHit$^+$, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$

FilterDeepHit$^+$ For the models with filter feature selection, random search also determines the number of top ranked features selected for each cause, $m_k, k = 1, \ldots, K$. Given the number of input features for the PLCO and SEER datasets is approx. 100, we let the model dynamically choose the optimal number of selected features from the choices in Table 5.6.

HybridDeepHit$^+$ In this model family we calculate mean importance and p-values based on $I = 20$ permutations of the validation dataset for each feature. The models HybridDeepHit$^+$ (M) and (P) are trained using the number of top features selected

Hyperparameter	Set of choices
$m_k, k = 1, \ldots, K$, number of selected features:	20, 40, 60

Table 5.6: Additional hyperparameter choices for FilterDeepHit$^+$ and HybridDeepHit$^+$ (M) and (P)

m_k. We set their random search sets identically to those for FilterDeepHit$^+$, as shown in Table 5.6. HybridDeepHit$^+$ (M/Cut) and (P/Cut) introduce different hyperparameters, cutoffs $c_k, k = 1, \ldots, K$ determining feature inclusion from mean importance values or p-values. We also determine these cutoffs using random search. The search sets are given in Table 5.7.

Hyperparameter	Set of choices
$c_k, k = 1, \ldots, K$, feature selection cutoffs:	0.0001, 0.001, 0.01

Table 5.7: Additional hyperparameter choices for HybridDeepHit$^+$ (M/Cut) and (P/Cut)

SparseDeepHit$^+$ Here there are additional regularisation hyperparameters to determine. We therefore adapt random search to additionally find the best values of $\gamma_k, k = 1, \ldots, K$, the relative weightings of the cause-specific regularisation terms. In order to reduce the search space, we set the weighting for the shared sub-network's regularisation term to be $\gamma_s = \frac{1}{K} \sum_{k=1}^{K} \gamma_k$. The sets from which the hyperparameters were chosen are given in Table 5.8.

Hyperparameter	Set of choices
$\gamma_k, k = 1, \ldots, K$, regularisation loss weights:	0.00001, 0.0001, 0.001

Table 5.8: Additional hyperparameter choices for SparseDeepHit$^+$

AttentiveDeepHit$^+$ The hyperparameters introduced here are regularisation loss weights γ_o and $\gamma_k, k = 1, \ldots, K$. Analogous to the case of SparseDeepHit$^+$, we adapt random search to include the γ_k, and fix $\gamma_o = \frac{1}{K} \sum_{k=1}^{K} \gamma_k$. The sets from which the hyperparameters were chosen are given in Table 5.9.

Hyperparameter	Set of choices
$\gamma_k, k = 1, \ldots, K$, regularisation loss weights:	0.00001, 0.0001, 0.001

Table 5.9: Additional hyperparameter choices for AttentiveDeepHit$^+$

5.3 Performance results and analysis

5.3.1 Benchmarks, DeepHit and DeepHit$^+$ on PLCO

	Event	Event 1		Event 2	
		Mean	SD.	Mean	SD.
Horizon	**Algorithm**				
$\Delta t = 012$	**RSF**	**0.934**	0.006	0.716	0.040
	DeepHit$^+$	0.927	0.009	**0.760**	0.030
	DeepHit	0.906	0.013	0.746	0.038
	Fine-Gray	0.710	0.020	**0.760**	0.028
	csCox	0.706	0.020	0.759	0.028
$\Delta t = 036$	**RSF**	**0.884**	0.006	0.742	0.016
	DeepHit$^+$	0.871	0.007	0.758	0.011
	DeepHit	0.835	0.010	0.746	0.012
	Fine-Gray	0.666	0.021	0.757	0.011
	csCox	0.665	0.016	**0.760**	0.012
$\Delta t = 060$	**RSF**	**0.866**	0.006	0.748	0.011
	DeepHit$^+$	0.845	0.008	0.756	0.012
	DeepHit	0.807	0.012	0.744	0.013
	Fine-Gray	0.647	0.013	0.755	0.013
	csCox	0.647	0.010	**0.758**	0.014
$\Delta t = 120$	**RSF**	**0.813**	0.005	0.737	0.007
	DeepHit$^+$	0.789	0.008	0.744	0.004
	DeepHit	0.749	0.011	0.726	0.007
	Fine-Gray	0.622	0.008	0.746	0.005
	csCox	0.621	0.008	**0.748**	0.005

Table 5.10: C-index performance of benchmark models, DeepHit and DeepHit$^+$ on PLCO

We present C-index results from cause-specific Cox models (csCox), Random Survival Forest (RSF), Fine-Gray, DeepHit and DeepHit$^+$ on the PLCO dataset for prostate cancer prediction in Table 5.10. It is generally observable that the best predictions for prostate cancer incidence (Event 1) attain a higher C-index (of around 0.8-0.9) than predictions of death incidence (Event 2). Heuristically, this is to be expected given the tailoring of features towards prostate cancer in the PLCO trial. However, with a C-index of around 0.75, the learning algorithms are able to obtain

significant information about Event 2 too, as compared to random guessing that would achieve a C-index of 0.5.

RSF best of traditional algorithms There is large variability in C-index performance for the traditional algorithms RSF, Fine-Gray and csCox (see Table 5.10). For prostate cancer incidence prediction (Event 1 on PLCO), RSF performs significantly better than the other algorithms across all evaluation times, with a C-index of 0.813 at 120 months, and 0.934 at 12 months evaluation horizon. Fine-Gray and csCox exhibit very similar performance due to their similarity in modelling the sub-distribution hazard and hazard function, respectively. However, the inherent linearity of these models constrains them to C-index scores around 0.2 below, and thus significantly worse, than RSF across all evaluation times. RSF can benefit from its non-parametric nature in order to exploit the non-linear relationship between the covariates and prostate cancer incidence.

All methods show much more similar performance for prediction of death (Event 2 on PLCO). With the exception of RSF at 12-months, the algorithms do not differ by more than 0.02. We note that the 12 month evaluation time should be treated with caution due to the small number of cases that can be used to test results (see Table 5.1). csCox performs best for all other evaluation times, indicating the existence of a more linear relationship between the covariates and death incidence.

DeepHit underperforms on PLCO Whilst the deep learning approach DeepHit is able to outperform traditional machine learning algorithms on the datasets in (Lee et al. 2018), Table 5.10 shows that its performance is lagging consistently behind RSF for prostate cancer incidence prediction (Event 1) on the real-world PLCO dataset. DeepHit is able to capture a significant portion of the non-linear relationships between features and outcome, but still trails by 0.03-0.06 in terms of C-index. Given that the C-index standard deviations are around and below 0.01 for the five cross-validation runs, this is a significant performance deficit. For death incidence prediction (Event 2), DeepHit is closer to the best performing method, but still lagging by 0.01-0.02.

DeepHit$^+$ with improvements, but still below RSF DeepHit's performance deficit is to a large part due to overfitting, as Table 5.10 shows DeepHit$^+$ achieves significant improvements stemming from modifications to layer sizing. As outlined in Section 4.1.4, DeepHit fixes the number of hidden nodes in each layer to 3, 5 and 3 times the covariate dimension, respectively. Given PLCO's large set of 105 features,

this results in a high number of parameters and thus the danger of overfitting. By conducting random search on both the number of layers and layer sizes, DeepHit$^+$ can find where reducing the network size combats overfitting and improves the learning. It is evident from the optimal hyperparameters (shown in Table 5.11) that a significant reduction in the network size occured: The number of nodes in each cause-specific layer were set to 50, whilst in DeepHit they were fixed as a multiple of the input dimension to 525 and 315 nodes, respectively.

Algorithm Parameter	Hyperparameter value DeepHit	 DeepHit$^+$
Weight of ranking loss, β	0.3	0.3
Parameter for risk comparator, σ	0.3	0.3
Minibatch size	050	050
Dropout keep-probability	0.6	0.6
Training step size	1E-04	1E-04
Shared: Nodes/layer, h_s	315	200
Shared: # layers, n_s	1	3
Cause-specific: Nodes/layer, h_k	525/315, 525/315	50, 50
Cause-specific: # layers, n_k	2	2, 1

Table 5.11: Hyperparameters for DeepHit and DeepHit$^+$ on PLCO

The improvement is reflected in C-index results 0.02 above DeepHit for 12 months evaluation horizon, and 0.04 above DeepHit for all other times on the main event of interest, prostate cancer incidence (Event 1). However, DeepHit$^+$ is unable to close the performance gap to the top, with a C-index still 0.01-0.02 below RSF. This motivates the need for further improvements on the deep learning architecture.

On Event 2, DeepHit$^+$'s improvements bring it to within at most 0.004 of the top performing method for each evaluation time.

5.3.2 Automatic feature selection on PLCO

We present the results on the PLCO dataset from all models, including Filter-DeepHit$^+$, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$, in Table 5.12.

5.3.2.1 FilterDeepHit$^+$

The results in Table 5.12 show that using feature selection with FilterDeepHit$^+$ achieves a significant performance benefit over DeepHit$^+$ for prostate cancer incidence

	Event	Event 1			Event 2		
		Mean	SD	Δ	Mean	SD	Δ
Horizon	**Algorithm**						
$\Delta t = 012$	**HybridDeepHit$^+$ (M)**	**0.936**	0.008	0.008	0.748	0.032	-0.012
	HybridDeepHit$^+$ (P)	0.935	0.004	0.008	0.753	0.033	-0.008
	HybridDeepHit$^+$ (P/Cut)	0.935	0.009	0.008	0.752	0.033	-0.008
	FilterDeepHit$^+$ (Anova)	0.935	0.009	0.007	0.753	0.028	-0.007
	RSF	0.934	0.006	0.007	0.716	0.040	-0.045
	HybridDeepHit$^+$ (M/Cut)	0.934	0.005	0.007	0.752	0.036	-0.009
	FilterDeepHit$^+$ (SVM)	0.933	0.007	0.005	0.741	0.050	-0.019
	FilterDeepHit$^+$ (ReliefF)	0.931	0.007	0.004	0.747	0.034	-0.013
	SparseDeepHit$^+$	0.930	0.008	0.003	**0.760**	0.027	-0.000
	DeepHit$^+$	0.927	0.009	0.000	**0.760**	0.030	0.000
	AttentiveDeepHit$^+$	0.917	0.010	-0.011	0.740	0.037	-0.020
	DeepHit	0.906	0.013	-0.021	0.746	0.038	-0.014
	Fine-Gray	0.710	0.020	-0.218	**0.760**	0.028	-0.001
	csCox	0.706	0.020	-0.221	0.759	0.028	-0.002
$\Delta t = 036$	**HybridDeepHit$^+$ (M)**	**0.890**	0.004	0.020	0.752	0.014	-0.006
	FilterDeepHit$^+$ (Anova)	0.888	0.006	0.017	0.755	0.015	-0.003
	HybridDeepHit$^+$ (M/Cut)	0.888	0.004	0.017	0.757	0.014	-0.001
	HybridDeepHit$^+$ (P)	0.888	0.005	0.017	0.750	0.011	-0.008
	FilterDeepHit$^+$ (SVM)	0.887	0.006	0.016	0.745	0.015	-0.013
	SparseDeepHit$^+$	0.886	0.005	0.016	0.758	0.011	0.000
	HybridDeepHit$^+$ (P/Cut)	0.886	0.003	0.015	0.756	0.013	-0.003
	FilterDeepHit$^+$ (ReliefF)	0.885	0.005	0.014	0.749	0.017	-0.009
	RSF	0.884	0.006	0.013	0.742	0.016	-0.016
	AttentiveDeepHit$^+$	0.876	0.008	0.005	0.744	0.009	-0.014
	DeepHit$^+$	0.871	0.007	0.000	0.758	0.011	0.000
	DeepHit	0.835	0.010	-0.036	0.746	0.012	-0.012
	Fine-Gray	0.666	0.021	-0.205	0.757	0.011	-0.001
	csCox	0.665	0.016	-0.205	**0.760**	0.012	0.002
$\Delta t = 060$	**HybridDeepHit$^+$ (M)**	**0.872**	0.006	0.027	0.751	0.015	-0.005
	FilterDeepHit$^+$ (Anova)	0.871	0.006	0.025	0.751	0.016	-0.005
	FilterDeepHit$^+$ (SVM)	0.870	0.004	0.025	0.744	0.017	-0.012
	HybridDeepHit$^+$ (M/Cut)	0.869	0.004	0.024	0.755	0.017	-0.001
	HybridDeepHit$^+$ (P)	0.869	0.003	0.024	0.747	0.012	-0.009
	SparseDeepHit$^+$	0.869	0.004	0.024	0.757	0.012	0.001
	FilterDeepHit$^+$ (ReliefF)	0.866	0.003	0.021	0.746	0.016	-0.010
	HybridDeepHit$^+$ (P/Cut)	0.866	0.006	0.021	0.753	0.015	-0.003
	RSF	0.866	0.006	0.020	0.748	0.011	-0.009
	AttentiveDeepHit$^+$	0.860	0.005	0.015	0.745	0.012	-0.012
	DeepHit$^+$	0.845	0.008	0.000	0.756	0.012	0.000
	DeepHit	0.807	0.012	-0.038	0.744	0.013	-0.013
	csCox	0.647	0.010	-0.198	**0.758**	0.014	0.002
	Fine-Gray	0.647	0.013	-0.198	0.755	0.013	-0.002
$\Delta t = 120$	**HybridDeepHit$^+$ (M)**	**0.821**	0.005	0.032	0.738	0.002	-0.006
	SparseDeepHit$^+$	0.820	0.006	0.031	0.745	0.005	0.001
	FilterDeepHit$^+$ (SVM)	0.820	0.006	0.031	0.733	0.006	-0.011
	HybridDeepHit$^+$ (M/Cut)	0.819	0.004	0.030	0.741	0.004	-0.003
	FilterDeepHit$^+$ (Anova)	0.819	0.007	0.030	0.739	0.003	-0.005
	HybridDeepHit$^+$ (P)	0.819	0.003	0.030	0.734	0.007	-0.010
	FilterDeepHit$^+$ (ReliefF)	0.814	0.006	0.025	0.732	0.004	-0.012
	HybridDeepHit$^+$ (P/Cut)	0.814	0.003	0.025	0.739	0.005	-0.005
	RSF	0.813	0.005	0.025	0.737	0.007	-0.007
	AttentiveDeepHit$^+$	0.811	0.004	0.023	0.734	0.007	-0.010
	DeepHit$^+$	0.789	0.008	0.000	0.744	0.004	0.000
	DeepHit	0.749	0.011	-0.039	0.726	0.007	-0.017
	Fine-Gray	0.622	0.008	-0.167	0.746	0.005	0.003
	csCox	0.621	0.008	-0.168	**0.748**	0.005	0.004

Table 5.12: C-index performance of all models on PLCO. SD denotes standard deviation, Δ denotes C-index difference to DeepHit$^+$.

(Event 1), whilst often underperforming for prediction of death incidence (Event 2). We exhibit their optimal hyperparameters from random search in Table 5.13.

	FilterDeepHit$^+$		
Algorithm	Anova	ReliefF	SVM
Parameter			
Weight of ranking loss, β	0.3	0.3	0.3
Parameter for risk comparator, σ	0.3	0.3	0.3
Minibatch size	050	050	050
Dropout keep-probability	0.6	0.6	0.6
Training step size	1E-04	1E-04	1E-04
Shared: Nodes/layer, h_s	50	100	100
Shared: # layers, n_s	2	2	2
Cause-specific: Nodes/layer, h_k	50, 50	50, 50	50, 50
Cause-specific: # layers, n_k	1, 1	1, 1	1, 2
# features selected , m_k	20, 20	40, 20	20, 60

Table 5.13: Optimal hyperparameters for the FilterDeepHit$^+$ family on PLCO

FilterDeepHit$^+$ (ANOVA) The best performing filter method is analysis of variance (ANOVA). On Event 1, it is able to outperform DeepHit$^+$ by a C-index difference of 0.007 to 0.030, significant for the already high performance level. It also beats the previously best non-deep learning benchmark RSF on all evaluation horizons. Its maximal C-index performance deficit against the top method for each time horizon is 0.002 for Event 1. On Event 2 it performs slightly worse than DeepHit$^+$, its C-index performance here lagging at most 0.009 behind the top algorithms from this study. However, given that on this secondary event the overall C-index is relatively low and the standard deviation often higher than for Event 1, we do not consider this performance difference to be material.

It is instructive to see in Table 5.13 that random search chooses only to use the top 20 features for both events, thus drastically reducing the features from the original number of 105. The improved performance shows the value of our feature selection. We present the top 20 ranked features used to train FilterDeepHit$^+$ (ANOVA) in Table 5.14. This table, as well as all other feature ranking tables in this section, exhibits results from the first of the five cross-validation train/test splits. The trial screening values *PSA level* and *abnormal, suspicious DRE result* are the top two ranked features and highly significant here. For Event 2, ANOVA gives a *heart attack* indicator, the number of *smoking pack years* (number of packs smoked per day times years smoked) and *age* as the most relevant factors.

Feature	Event 1
PSALevel	7.5e-154
DRE_AbnormSuspi	9.6e-47
DRE_Negative	1.7e-13
Age	3.0e-13
NumRelativesPrCancer	2.1e-06
FamHistProsCan_No	3.5e-06
FamHistProsCan_Yes	8.0e-06
Occ_Retired	1.4e-04
Occ_Working	3.2e-04
Race_BlackNonHisp	3.7e-04
DRE_NotDoneExp	4.6e-04
AgeUrinateAtNight	6.8e-04
HadProstateBiopsy	9.2e-04
ProsSurgeries_No	1.1e-03
ProsSurgeries_Yes	1.0e-02
FamHistCancer	1.7e-02
BMIAt50	2.6e-02
Race_Asian	3.0e-02
WeightAt50	3.0e-02
AgeEnlargedProstate	3.1e-02

Feature	Event 2
HeartAttack	2.2e-13
SmokingPackYears	4.5e-10
Age	1.6e-09
UrinateNight_3	2.2e-09
DurationSmokedCig	3.6e-09
AgeStoppedSmoking	1.3e-08
Marital_Married	3.8e-08
Emphysema	7.9e-08
Occ_Working	9.1e-08
DRE_NotDoneExp	1.4e-07
Occ_Disabled	2.1e-07
Stroke	3.1e-06
Marital_Divorced	3.9e-06
CigarettesPerDay	9.7e-06
YearsStoppedSmoking	2.1e-05
Occ_Retired	6.2e-05
Diabetes	1.1e-04
Hypertension	1.1e-04
Cig_Never	7.5e-04
EverSmoked	7.5e-04

Table 5.14: Top 20 features by ANOVA p-value on PLCO

FilterDeepHit$^+$ (SVM) The second best filter method of those that we tested is SVM, being outperformed by ANOVA in the first three time horizons for Event 1. However, on Event 2, FilterDeepHit$^+$ (SVM) is one of the worst performing methods, even though the significance of performance differences for this event are relatively small. We may see a potential reason for this in SVM's feature ranking (Table 5.15): Similarly to ANOVA, PSA level is seen as by far the most relevant feature for Event 1, although the second part of screening, an abnormal and suspicious DRE result, is given the 12th highest weight. For Event 2, all feature weights are extremely low, below 0.0001. This indicates a potential difficulty for SVMs to capture feature relationships in death incidence prediction. It may also be the reason that random search on the hyperparameters selects the largest number of features, 60, for Event 2 (Table 5.13).

FilterDeepHit$^+$ (ReliefF) Whilst still outperforming DeepHit$^+$ by a significant margin on Event 1, ReliefF performs the worst out of the three filter methods we tested. On Event 2, it shows a consistent performance deficit of up to 0.013 against DeepHit$^+$. The feature ranking obtained from ReliefF (shown in Table 5.16), also shows marked differences to ANOVA: PSA level is not captured as the outstanding

Feature	Event 1
PSALevel	0.3699
HadProstateBiopsy	0.0240
BMI	0.0219
WeightAt50	0.0189
Height	0.0172
Weight	0.0165
EnlargedProstateBPH	0.0144
DRE_AbnormSuspi	0.0123
DurationSmokedCig	0.0115
Occ_ExtSickLeave	0.0108
DRE_NotDoneExp	0.0105
DRE_AbnormNonSuspi	0.0101
NumRelativesPrCancer	0.0073
Age	0.0070
WeightAt20	0.0069
YearsStoppedSmoking	0.0063
Race_Asian	0.0062
SmokingPackYears	0.0059
BMIAt50	0.0058
AgeStoppedSmoking	0.0052

Feature	Event 2
Weight	0.0001
BMI	0.0001
YearsStoppedSmoking	0.0001
DurationSmokedCig	0.0001
BMIAt20	0.0001
WeightAt20	0.0001
PSALevel	0.0001
AgeStoppedSmoking	0.0001
ReturnedBQ	0.0001
WeightAt50	0.0001
Race_Missing	0.0000
BMIAt50	0.0000
Cig_Former	0.0000
Age	0.0000
SmokingPackYears	0.0000
CigarettesPerDay	0.0000
Cig_Never	0.0000
EverSmoked	0.0000
NumRelativesPrCancer	0.0000
Cig_Current	0.0000

Table 5.15: Top 20 features by SVM weights on PLCO

feature for Event 1, though it appears in the top group mixed with age, height and weight features. The (abnormal, suspicious) DRE result is ranked as 9th most relevant feature. For Event 2, the heart attack indicator is ranked 16th, down from 1st in ANOVA.

5.3.2.2 HybridDeepHit$^+$

We also benchmark the performance results of the HybridDeepHit$^+$ family of models on the PLCO dataset in Table 5.12, and give their optimal hyperparameters in Table 5.17.

HybridDeepHit$^+$ (M) and (M/Cut) Our hybrid feature selection model with mean importance using (HybridDeepHit$^+$ (M)) is the best performing model at all evaluation horizons for Event 1, achieving improvements over DeepHit$^+$ by a C-index of 0.008 ($\Delta t = 12$) to 0.032 ($\Delta t = 120$). It also improves on RSF by 0.002 to 0,008. The model's relative performance is lower on Event 2, yet as previously explained, this difference is less significant.

HybridDeepHit$^+$ (M/Cut), which uses importance cutoffs instead of selecting a certain number of features, is not quite as successful for Event 1, and performs 0.002-

Feature	Event 1
Age	0.1742
Height	0.1673
PSALevel	0.1556
WeightAt20	0.1530
BMIAt20	0.1370
WeightAt50	0.1340
Weight	0.1299
BMI	0.1290
DRE_AbnormSuspi	0.1243
AgeStoppedSmoking	0.1215
Cig_Former	0.1211
BMIAt50	0.1192
CigarettesPerDay	0.0979
YearsStoppedSmoking	0.0967
DurationSmokedCig	0.0713
Cig_Never	0.0670
EverSmoked	0.0670
NumRelativesPrCancer	0.0655
Occ_Retired	0.0646
Cig_Current	0.0591

Feature	Event 2
Age	0.2641
Height	0.2023
WeightAt20	0.1941
Cig_Former	0.1935
WeightAt50	0.1877
Weight	0.1829
AgeStoppedSmoking	0.1814
BMI	0.1748
CigarettesPerDay	0.1748
BMIAt20	0.1723
BMIAt50	0.1699
YearsStoppedSmoking	0.1690
DurationSmokedCig	0.1524
Cig_Never	0.1401
EverSmoked	0.1401
HeartAttack	0.1058
Occ_Retired	0.1044
SmokingPackYears	0.1040
AgeUrinateAtNight	0.1026
DRE_NotDoneExp	0.0972

Table 5.16: Top 20 features by ReliefF on PLCO

	HybridDeepHit$^+$			
Algorithm Parameter	M	M/Cut	P	P/Cut
Weight of ranking loss, β	0.3	0.3	0.3	0.3
Parameter for risk comparator, σ	0.3	0.3	0.3	0.3
Minibatch size	050	050	050	050
Dropout keep-probability	0.6	0.6	0.6	0.6
Training step size	1E-04	1E-04	1E-04	1E-04
Shared: Nodes/layer, h_s	50	100	50	50
Shared: # layers, n_s	2	2	2	2
Cause-specific: Nodes/layer, h_k	50, 50	100, 100	50, 50	100, 100
Cause-specific: # layers, n_k	1, 1	2, 2	1, 1	2, 2
# features selected , m_k	20, 20	-	20, 20	-
Feature selection cutoff, c_k	-	0.001, 0.001	-	0.001, 0.0001

Table 5.17: Optimal hyperparameters for the HybridDeepHit$^+$ family on PLCO

0.003 behind HybridDeepHit$^+$ (M). For the secondary Event 2, (M/Cut) outperforms (M) by a C-index difference of 0.003-0.005. Whilst both methods should in theory yield equivalent results when all possibilities for hyperparameters are explored, by choosing specific sets of potential hyperparameters for random search we restrict each model. In practice, setting the right cutoff c_k may be difficult, as it is unclear

at which order of magnitude to include features. Our search set (Table 5.7) includes three values that differ by a factor of 10, potentially too large a mesh to find optimal choices for Event 1.

Apart from two time horizons on Event 2, where FilterDeepHit$^+$ achieves a slightly higher C-index, HybridDeepHit$^+$ (M) outperforms all filter methods for every event and evaluation time. This indicates there is particular value in extracting feature importances with a hybrid method that can capture interactions with the model, and potentially more accurate feature-label relationships with a neural network. We show the top 20 features by mean importance values as calculated on the first train/test split's validation dataset in Table 5.18. Just like for the high-performing filter method ANOVA, PSA level and DRE result rank as the top two features for Event 1, and age, heart attack and pack years as the top three for Event 2.

Event 1	Mean	p-value
PSALevel	0.2692	7.2e-28
DRE_AbnormSuspi	0.0085	5.4e-21
Cig_Former	0.0071	4.8e-13
DRE_NotDoneExp	0.0037	5.1e-17
DurationSmokedCig	0.0037	3.4e-14
Race_Asian	0.0029	1.6e-06
HadProstateBiopsy	0.0028	5.8e-13
DRE_Negative	0.0026	6.2e-13
Race_BlackNonHisp	0.0022	1.5e-12
ProsSurgeries_No	0.0020	1.1e-11
SmokeNow	0.0018	7.9e-07
ProsSurgeries_Unkn	0.0018	3.6e-07
Occ_Working	0.0017	1.3e-07
ProstateProblem	0.0016	3.7e-07
ProsSurgeries_Yes	0.0016	3.3e-10
BMIAt50	0.0014	4.8e-09
Emphysema	0.0013	2.6e-11
Aspirin	0.0013	3.6e-07
Marital_Divorced	0.0012	8.5e-10
ProsTransuResection	0.0011	1.1e-07

Event 2	Mean	p-value
Age	0.0554	1.2e-17
HeartAttack	0.0254	1.2e-18
SmokingPackYears	0.0068	2.7e-11
Diabetes	0.0051	9.9e-09
DRE_NotDoneExp	0.0047	1.5e-12
DurationSmokedCig	0.0038	1.2e-10
Occ_Disabled	0.0035	1.4e-09
AgeStoppedSmoking	0.0028	6.9e-11
ReturnedBQ	0.0026	4.5e-11
PersCanHist_Yes	0.0023	8.4e-09
Hypertension	0.0022	2.7e-07
UrinateNight_1	0.0021	3.8e-09
Emphysema	0.0020	2.9e-07
Stroke	0.0019	1.5e-06
NumIbuprofen	0.0018	4.9e-07
Cig_Never	0.0017	1.5e-06
PSALevel	0.0016	7.2e-03
SmokeNow	0.0015	9.8e-04
CigarettesPerDay	0.0014	1.1e-09
FamHistProsCan_Yes	0.0014	1.2e-08

Table 5.18: Top 20 features by mean importance for DeepHit$^+$ on PLCO

HybridDeepHit$^+$ (P) and (P/Cut) In most cases, using the constructed importance p-value for feature selection only achieves a marginally lower C-index than mean value models. On Event 1, HybridDeepHit$^+$ (P) performs 0.001-0.003 below (M), whilst on Event 2 it performs better for $\Delta t = 12$, but worse for all other evaluation times. This may be due to the fact that some features (such as *stroke* for Event

1) with highly significant p-values can still have lower mean importance, as shown in Table 5.19. Heuristically, leaving out features with very low importance values may be beneficial, even if the importance is positive, if the model is then able to focus on more impactful features. Just as before, we also see the (P/Cut) model performing worse than (P) on Event 1, but marginally better on Event 2.

When considering whether to choose the mean or p-value to conduct feature selection, there are also additional considerations to make. Whilst mean value models may show better performance on this dataset, there could conceivably be examples where a rare medical condition captured in a feature is highly significant for prediction, but occurs too infrequently to have a large enough impact on the overall model C-index. This would result in a low mean importance value, and care must thus be taken when choosing the features to exclude.

Event 1	p-value	Mean
PSALevel	7.2e-28	0.2692
DRE_AbnormSuspi	5.4e-21	0.0085
DRE_NotDoneExp	5.1e-17	0.0037
DurationSmokedCig	3.4e-14	0.0037
Cig_Former	4.8e-13	0.0071
DRE_Negative	6.2e-13	0.0026
HadProstateBiopsy	5.8e-13	0.0028
Race_BlackNonHisp	1.5e-12	0.0022
ProsSurgeries_No	1.1e-11	0.0020
Emphysema	2.6e-11	0.0013
ProstateSurgeryAge	1.0e-10	0.0010
ProsSurgeries_Yes	3.3e-10	0.0016
Marital_Divorced	8.5e-10	0.0012
BMIAt50	4.8e-09	0.0014
Stroke	1.2e-08	0.0006
BMI	4.5e-08	0.0011
Race_WhiteNonHisp	9.1e-08	0.0009
ProsTransuResection	1.1e-07	0.0011
Occ_Working	1.3e-07	0.0017
ProsSurgeries_Unkn	3.6e-07	0.0018

Event 2	p-value	Mean
HeartAttack	1.2e-18	0.0254
Age	1.2e-17	0.0554
Race_Missing	4.8e-13	0.0009
DRE_NotDoneExp	1.5e-12	0.0047
SmokingPackYears	2.7e-11	0.0068
ReturnedBQ	4.5e-11	0.0026
AgeStoppedSmoking	6.9e-11	0.0028
DurationSmokedCig	1.2e-10	0.0038
Occ_Other	7.5e-10	0.0009
CigarettesPerDay	1.1e-09	0.0014
Occ_Unempl	1.1e-09	0.0013
Occ_Disabled	1.4e-09	0.0035
Ibuprofen	2.1e-09	0.0014
NumRelativesPrCancer	2.5e-09	0.0009
UrinateNight_1	3.8e-09	0.0021
PersCanHist_Yes	8.4e-09	0.0023
Diabetes	9.9e-09	0.0051
FamHistProsCan_Yes	1.2e-08	0.0014
Aspirin	2.1e-08	0.0011
Hypertension	2.7e-07	0.0022

Table 5.19: Top 20 features by importance p-value for DeepHit$^+$ on PLCO

5.3.2.3 SparseDeepHit$^+$

SparseDeepHit$^+$ consistently performs within 0.006 C-index of the best method in our PLCO test (see Table 5.12). The method is very consistent across events, performing significantly better than DeepHit$^+$ for Event 1, and achieving equal or better performance on Event 2 for all time horizons. On Event 1, its relative performance

improves for larger time horizons, performing just 0.001 behind HybridDeepHit$^+$ at $\Delta t = 120$.

The model's architecture is specifically designed to learn weights for input features to the shared and cause-specific sub-networks. Given these weights are regularized, the model will use large weights only when necessary to improve predictive value. The weights therefore also represent a view on feature relevance. We present the top 20 for each sub-network in Table 5.20, denoting input features as *Shared*** (** a placeholder for the node number) when a cause-specific sub-network utilizes a feature from the shared connection $f_s(\mathbf{x})$. It can be seen that whilst PSA level and age at trial entry are dominating, further analysis of the weights is difficult as shared features are highly ranked, but not immediately attributable to original features from the dataset.

Shared		Event 1		Event 2	
Feature		Feature		Feature	
PSALevel	0.5506	PSALevel	-0.5766	PSALevel	0.7646
EnlargedProsBPH	0.1409	Age	0.2433	Age	0.3417
Marital_Married	0.1373	Shared20	0.2011	DRE_AbnormSuspi	0.2718
AgeInflamedPros	-0.1267	Shared49	0.1979	PersCanHist_No	-0.2393
UrinateNight_Never	0.1223	Shared42	-0.1919	Shared29	-0.2379
Prostatectomy	-0.1193	Shared38	0.1913	Shared42	0.2371
Marital_NeverMarr	-0.1169	Shared25	0.1854	DurationSmokedCig	0.2338
PersCanHist_Unkn	-0.1161	Cig_Current	-0.1821	Shared35	0.2133
Hypertension	0.1135	Hypertension	-0.1726	Occ_Disabled	0.2043
GallbladderStones	-0.1114	HeartAttack	0.1725	SmokingPackYears	0.2041
InflamedProstate	0.1094	Shared14	0.1708	Shared48	-0.1983
Brothers	-0.1083	Shared22	-0.1689	FamHistPrCan_Yes	-0.1920
Occ_Working	-0.1028	Shared1	0.1648	SmokeNow	-0.1904
Sisters	-0.1005	Diabetes	-0.1647	Diabetes	-0.1873
Pipe_Former	-0.0973	Education	0.1641	HeartAttack	-0.1869
FamHistPrCan_Pos	-0.0959	Shared29	-0.1622	FamHistPrCan_No	0.1817
Cigar_Former	-0.0947	UrinateNight_1	0.1595	Emphysema	0.1775
Emphysema	-0.0927	Shared5	0.1571	AgeEnlargedProstate	0.1774
BMI	-0.0926	Shared26	-0.1561	EverSmoked	-0.1704
DurationSmokedCig	-0.0905	Race_Asian	0.1555	CigarettesPerDay	-0.1657

Table 5.20: Top 20 features by first layer weights in SparseDeepHit$^+$'s sub-networks on PLCO. The ranking is determined by the weight's absolute value.

5.3.2.4 AttentiveDeepHit$^+$

Of the automatic feature selection methods introduced in this work, AttentiveDeepHit$^+$ shows the smallest gain over DeepHit$^+$ on Event 1, and is unable to outperform - DeepHit$^+$ for $\Delta t = 12$. Furthermore, for Event 2, the method was one of the worst performing of all models. This may be because the regularisation parameters are not

well enough optimized: In Table 5.21 we see that random search determined a scaling factor of 10^{-5} to be optimal, the smallest value given in the sets for random search. Using even smaller values, or maybe even no regularisation (despite the theory), may yield a performance improvement.

Algorithm Parameter	Hyperparameter value SparseDeepHit$^+$	 AttentiveDeepHit$^+$
Weight of ranking loss, β	0.3	0.3
Regularization scale, γ_k	1e-05, 0.001	1e-05, 1e-05
Parameter for risk comparator, σ	0.3	0.3
Minibatch size	050	050
Dropout keep-probability	0.6	0.6
Training step size	1E-04	1E-04
Shared: Nodes/layer, h_s	50	50
Shared: # layers, n_s	2	2
Cause-specific: Nodes/layer, h_k	100, 100	50, 100
Cause-specific: # layers, n_k	2, 1	2, 1

Table 5.21: Optimal hyperparameters for SparseDeepHit$^+$ and AttentiveDeepHit$^+$ on PLCO

We are, however, able to extract meaningful feature rankings from the attention sub-network by considering the values of its output nodes that are used as the attention mask. These values are patient-specific, as they are calculated by the attention network depending on the input features. In Table 5.22, we report the average values for the top 20 features across all patients in the test dataset of the first train/test split. We see PSA level and age dominate for Events 1 and 2, respectively, similar to the rankings given by the other feature selection methods in this work. These values, however, vary strongly from patient to patient, and therefore could also be used to enhance patient-specific interpretability of on what basis the model made its prediction.

5.3.3 SEER results: The impact of a large feature set

Table 5.23 shows the performance of DeepHit when applied to the SEER and SEER-synth datasets. Across the board, performance on the synthetic dataset (SEERsynth) is worse than on the original dataset. For our main event of interest, breast cancer incidence prediction (Event 1), the C-index performance gap of DeepHit between the two datasets is as big as 0.09. The SEERsynth dataset is identical to the original SEER dataset bar 100 additionally generated unrelated features. It therefore appears

Feature	Event 1
PSALevel	0.7200
Age	0.0473
Filt_Filtered	0.0215
Hypertension	0.0171
Cig_Current	0.0147
HeartAttack	0.0128
Cig_Former	0.0120
AgeStoppedSmoking	0.0100
Diabetes	0.0098
Filt_NonFiltered	0.0081
BMIAt20	0.0079
Vasectomy	0.0075
Race_WhiteNonHisp	0.0064
Race_BlackNonHisp	0.0061
EnlargedProsBPH	0.0059
Race_Asian	0.0055
WeightAt50	0.0054
CigarettesPerDay	0.0052
DurationSmokedCig	0.0051
UrinateNight_1	0.0050

Feature	Event 2
Age	0.2448
EverSmoked	0.1677
Occ_Working	0.0681
HeartAttack	0.0584
DurationSmokedCig	0.0546
Diabetes	0.0370
UrinateNight_Never	0.0329
Hypertension	0.0278
YearsStoppedSmoking	0.0278
NumAspirin	0.0276
DRE_AbnormSuspi	0.0267
Cig_Current	0.0253
DRE_NotDoneExp	0.0209
SmokingPackYears	0.0198
FamHistPrCan_Yes	0.0168
Ibuprofen	0.0154
BMIAt20	0.0119
UrinateNight_1	0.0090
AgeStoppedSmoking	0.0090
Stroke	0.0082

Table 5.22: Top 20 features by average attention values in AttentiveDeepHit$^+$ on PLCO

that the noisy additional features are the cause for the loss in DeepHit's performance between the two datasets.

A similar pattern is evident in DeepHit$^+$. Even after optimizing the network size in order to combat overfitting, a C-index performance gap between the two datasets largely persists. Whilst DeepHit$^+$ is able to achieve significant performance gains of up to 0.02 on SEERsynth, the performance of DeepHit and DeepHit$^+$ on SEER is very similar. Thus with the lower number of input features in SEER, DeepHit's neural network is already able to learn well without excessive overfitting. These experimental results give additional motivation for the value of feature selection prior to training a deep neural network when there are a large number of potentially irrelevant features.

We provide results of all methods on the SEERsynth dataset in Appendix A, Table A.2.

	Event	Event 1		Event 2		Event 3	
		Mean	SD	Mean	SD	Mean	SD
Horizon	**Algorithm**						
$\Delta t = 012$	**DeepHit (SEER)**	**0.854**	0.015	0.674	0.097	0.771	0.019
	DeepHit$^+$ (SEER)	0.853	0.016	0.676	0.078	**0.775**	0.021
	DeepHit$^+$ (SEERsynth)	0.822	0.017	**0.687**	0.057	0.750	0.019
	DeepHit (SEERsynth)	0.798	0.021	0.660	0.048	0.736	0.024
$\Delta t = 036$	**DeepHit (SEER)**	**0.796**	0.009	**0.689**	0.036	0.720	0.018
	DeepHit$^+$ (SEER)	**0.796**	0.009	0.688	0.037	**0.721**	0.020
	DeepHit$^+$ (SEERsynth)	0.749	0.006	0.671	0.016	0.697	0.023
	DeepHit (SEERsynth)	0.725	0.008	0.652	0.036	0.697	0.029
$\Delta t = 060$	**DeepHit (SEER)**	**0.773**	0.004	0.679	0.038	**0.704**	0.013
	DeepHit$^+$ (SEER)	0.772	0.004	**0.682**	0.045	**0.704**	0.014
	DeepHit$^+$ (SEERsynth)	0.717	0.007	0.665	0.008	0.682	0.014
	DeepHit (SEERsynth)	0.695	0.010	0.648	0.033	0.682	0.014
$\Delta t = 120$	**DeepHit (SEER)**	**0.748**	0.002	**0.659**	0.026	**0.704**	0.010
	DeepHit$^+$ (SEER)	**0.748**	0.002	0.657	0.031	0.703	0.011
	DeepHit$^+$ (SEERsynth)	0.672	0.004	0.634	0.015	0.683	0.009
	DeepHit (SEERsynth)	0.661	0.010	0.618	0.021	0.684	0.010

Table 5.23: C-index performance of DeepHit and DeepHit$^+$ on SEER and SEERsynth

Chapter 6

Conclusion

In this dissertation we have exhibited novel methods for survival analysis that on real-world and synthetic datasets have achieved significant performance gains. In particular, we summarize and evaluate our main results below.

- In settings where the number of potentially irrelevant features is large, we gave evidence that the performance of deep learning survival analysis methods can significantly suffer. We presented the real-world PLCO dataset for which this is the case, and confirmed these results with a combination of real-world and synthetic data in the SEERsynth dataset.

- We made improvements to the previously cutting-edge deep learning survival model DeepHit by adapting its architecture and training procedure to create the model DeepHit$^+$. This model significantly outperformed DeepHit on the datasets.

- We subsequently proposed new models with automatic feature selection in deep survival analysis to further improve on DeepHit$^+$. To achieve this, we developed a new architecture that allows feature selection for multitask learning settings such as survival analysis with competing risks. We devised an adaptation to render commonly used filter feature selection techniques applicable to survival analysis, resulting in the model family FilterDeepHit$^+$. We applied a technique for extracting feature importance in new contexts of deep learning and survival analysis for our embedded feature selection method HybridDeepHit$^+$. Finally, with SparseDeepHit$^+$ and AttentiveDeepHit$^+$ we adapted two soft feature selection concepts to our context.

- With our experiments we gave a detailed and rigorous performance analysis, benchmarking all methods on the same dataset and a task of particular medical relevance.

- Whilst previous literature on feature selection for deep learning often focuses on one specific method, this is one of the first works to enable comparability of various filter, hybrid and soft feature selection methods. The success of our methods adds to the currently sparse literature on concrete improvements in this field.

- With the output of model-derived feature rankings, HybridDeepHit$^+$, SparseDeepHit$^+$ and AttentiveDeepHit$^+$ take a step to improve model interpretability, giving valuable insight into the features used for prediction.

More generally, we see our broader contribution as mainly threefold.

1. We give strong empirical reasons to increase focus on settings where a large feature space hinders performance, and believe this can be beneficial to many practical settings in the future.

2. Our novel architectures, particularly HybridDeepHit$^+$, show ways to improve deep learning models for survival analysis.

3. Finally, we have created methodologies that can be readily adapted to many other settings in deep learning outside survival analysis, taking a step to improve deep learning in tasks where datasets are insufficiently large for their number of features to train the model well.

With this encouragement in mind, there are also limitations of our work that can be starting points for future research. Firstly, in this dissertation we used two datasets to evaluate performance. More experimental validation would add robustness to the conclusions of this work. Secondly, we should strive for a better understanding of how to make the best feature selection from a given ranking. Whilst we used the number of features or cutoffs as hyperparameters in random search, this can be inefficient and even ineffective if not enough computational resources are available. A large step forward would be to create better guidance to improve the search for such feature cutoffs, or to set them without the need for search.

Given the vast breadth of existing techniques for feature selection, we have only been able to consider a subselection in this dissertation. With filter feature selection

methods, we would encourage more benchmarks for FilterDeepHit$^+$. For embedded methods, it would be instructive to compare performance of existing weight-based approaches with the importance-based approach of HybridDeepHit$^+$. We see our soft feature selection methods as particularly advantageous, as no retraining of the model is required and the mechanism is embedded into the network architecture. More research is required to improve their performance, especially in optimising their regularisation techniques. Finally, also other techniques of dimensionality reduction, including feature projection and embedding methods could be benchmarked.

Whilst all of the above is applicable to survival analysis, we also see potential for other deep learning methods with analogous issues, e.g. in classification or regression problems, to build upon the methodologies developed in this work. As we have shown, this research topic addresses real-world problems, and significant advances can be made.

Appendix A

Additional results

PLCO We present the mean performance of all methods across all evaluation horizons and events in Table A.1.

Algorithm	Mean C-index
SparseDeepHit$^+$	0.816
HybridDeepHit$^+$ (M/Cut)	0.814
FilterDeepHit$^+$ (Anova)	0.814
HybridDeepHit$^+$ (M)	0.814
HybridDeepHit$^+$ (P/Cut)	0.813
HybridDeepHit$^+$ (P)	0.812
FilterDeepHit$^+$ (SVM)	0.809
FilterDeepHit$^+$ (ReliefF)	0.809
DeepHit$^+$	0.806
RSF	0.805
AttentiveDeepHit$^+$	0.803
DeepHit	0.782
csCox	0.708
Fine-Gray	0.708

Table A.1: Average C-index performance of all models across all evaluation horizons and events on PLCO

SEERsynth We present results of all methods on the SEERsynth dataset in Table A.2. These were obtained using the experimental settings outlined in Chapter 5. Furthermore, we exhibit the mean performance of all methods across all evaluation horizons and events in Table A.3.

	Event	Event 1			Event 2			Event 3		
		Mean	SD	Δ	Mean	SD	Δ	Mean	SD	Δ
Horizon	**Algorithm**									
$\Delta t = 012$	**RSF**	**0.852**	0.015	0.029	0.616	0.081	-0.071	0.747	0.046	-0.003
	HybridDeepHit$^+$ (M)	0.848	0.016	0.026	0.692	0.072	0.005	0.763	0.023	0.013
	HybridDeepHit$^+$ (M/Cut)	0.847	0.013	0.025	0.684	0.065	-0.002	0.766	0.019	0.016
	SparseDeepHit$^+$	0.846	0.022	0.024	0.698	0.089	0.011	0.763	0.029	0.013
	AttentiveDeepHit$^+$	0.846	0.014	0.023	0.644	0.060	-0.043	0.761	0.025	0.011
	HybridDeepHit$^+$ (P/Cut)	0.844	0.015	0.022	**0.710**	0.071	0.023	0.763	0.024	0.013
	FilterDeepHit$^+$ (Anova)	0.840	0.018	0.018	0.662	0.124	-0.024	0.760	0.026	0.010
	HybridDeepHit$^+$ (P)	0.839	0.016	0.017	0.699	0.080	0.012	0.761	0.022	0.010
	FilterDeepHit$^+$ (ReliefF)	0.836	0.012	0.014	0.709	0.084	0.022	**0.769**	0.018	0.019
	DeepHit$^+$	0.822	0.017	0.000	0.687	0.057	0.000	0.750	0.019	0.000
	FilterDeepHit$^+$ (SVM)	0.809	0.045	-0.013	0.696	0.094	0.009	0.748	0.030	-0.003
	DeepHit	0.798	0.021	-0.025	0.660	0.048	-0.027	0.736	0.024	-0.014
	Fine-Gray	0.793	0.020	-0.029	0.642	0.061	-0.045	0.701	0.015	-0.050
	csCox	0.793	0.019	-0.029	0.650	0.081	-0.037	0.738	0.024	-0.012
$\Delta t = 036$	**RSF**	**0.792**	0.013	0.043	0.662	0.042	-0.009	0.704	0.024	0.007
	HybridDeepHit$^+$ (M)	0.786	0.012	0.037	0.690	0.036	0.019	0.714	0.022	0.016
	HybridDeepHit$^+$ (M/Cut)	0.786	0.007	0.037	0.680	0.027	0.008	**0.717**	0.021	0.020
	AttentiveDeepHit$^+$	0.783	0.007	0.035	0.624	0.050	-0.047	0.710	0.022	0.012
	SparseDeepHit$^+$	0.782	0.024	0.034	0.686	0.024	0.015	0.715	0.021	0.017
	HybridDeepHit$^+$ (P/Cut)	0.782	0.012	0.033	0.686	0.017	0.015	0.714	0.021	0.017
	FilterDeepHit$^+$ (Anova)	0.777	0.013	0.029	0.647	0.049	-0.025	0.715	0.021	0.017
	HybridDeepHit$^+$ (P)	0.776	0.011	0.027	0.688	0.021	0.017	0.707	0.020	0.009
	FilterDeepHit$^+$ (ReliefF)	0.773	0.026	0.024	**0.702**	0.024	0.031	0.714	0.019	0.017
	FilterDeepHit$^+$ (SVM)	0.761	0.020	0.013	0.695	0.022	0.024	0.699	0.029	0.002
	DeepHit$^+$	0.749	0.006	0.000	0.671	0.016	0.000	0.697	0.023	0.000
	DeepHit	0.725	0.008	-0.024	0.652	0.036	-0.019	0.697	0.029	-0.000
	csCox	0.699	0.009	-0.050	0.648	0.018	-0.023	0.697	0.015	-0.000
	Fine-Gray	0.697	0.009	-0.052	0.654	0.021	-0.017	0.674	0.013	-0.023
$\Delta t = 060$	**RSF**	**0.770**	0.006	0.053	0.680	0.041	0.015	0.693	0.016	0.011
	HybridDeepHit$^+$ (M)	0.763	0.008	0.046	0.682	0.031	0.018	0.695	0.012	0.013
	HybridDeepHit$^+$ (M/Cut)	0.761	0.010	0.044	0.674	0.023	0.009	**0.699**	0.010	0.017
	AttentiveDeepHit$^+$	0.760	0.004	0.043	0.620	0.040	-0.045	0.693	0.012	0.011
	SparseDeepHit$^+$	0.758	0.020	0.041	0.678	0.032	0.014	0.697	0.013	0.015
	HybridDeepHit$^+$ (P/Cut)	0.757	0.008	0.040	0.682	0.025	0.018	0.695	0.013	0.013
	FilterDeepHit$^+$ (Anova)	0.754	0.007	0.037	0.651	0.036	-0.014	0.697	0.013	0.016
	HybridDeepHit$^+$ (P)	0.750	0.007	0.033	0.687	0.018	0.022	0.691	0.009	0.009
	FilterDeepHit$^+$ (ReliefF)	0.749	0.028	0.032	**0.698**	0.022	0.033	0.696	0.014	0.014
	FilterDeepHit$^+$ (SVM)	0.741	0.016	0.024	0.695	0.029	0.030	0.685	0.014	0.003
	DeepHit$^+$	0.717	0.007	0.000	0.665	0.008	0.000	0.682	0.014	0.000
	DeepHit	0.695	0.010	-0.022	0.648	0.033	-0.016	0.682	0.014	-0.000
	csCox	0.669	0.007	-0.048	0.655	0.026	-0.010	0.688	0.011	0.006
	Fine-Gray	0.667	0.007	-0.050	0.662	0.035	-0.002	0.672	0.011	-0.010
$\Delta t = 120$	**RSF**	**0.746**	0.004	0.074	**0.670**	0.031	0.036	**0.697**	0.012	0.015
	FilterDeepHit$^+$ (Anova)	0.729	0.003	0.057	0.635	0.028	0.001	0.693	0.010	0.010
	AttentiveDeepHit$^+$	0.727	0.004	0.055	0.611	0.028	-0.023	0.692	0.010	0.009
	HybridDeepHit$^+$ (M)	0.724	0.009	0.052	0.658	0.026	0.024	0.688	0.010	0.005
	HybridDeepHit$^+$ (M/Cut)	0.719	0.011	0.047	0.649	0.023	0.015	0.692	0.010	0.009
	SparseDeepHit$^+$	0.719	0.013	0.047	0.652	0.032	0.018	0.696	0.009	0.014
	HybridDeepHit$^+$ (P/Cut)	0.712	0.011	0.040	0.658	0.023	0.024	0.689	0.010	0.007
	FilterDeepHit$^+$ (ReliefF)	0.706	0.029	0.034	0.665	0.019	0.031	0.692	0.010	0.009
	HybridDeepHit$^+$ (P)	0.705	0.008	0.033	0.658	0.017	0.024	0.689	0.008	0.006
	FilterDeepHit$^+$ (SVM)	0.701	0.011	0.029	0.660	0.023	0.026	0.685	0.009	0.003
	DeepHit$^+$	0.672	0.004	0.000	0.634	0.015	0.000	0.683	0.009	0.000
	DeepHit	0.661	0.010	-0.011	0.618	0.021	-0.016	0.684	0.010	0.002
	csCox	0.638	0.004	-0.034	0.631	0.018	-0.003	0.694	0.011	0.012
	Fine-Gray	0.637	0.005	-0.036	0.641	0.024	0.007	0.686	0.013	0.004

Table A.2: C-index performance of all models on SEERsynth. SD denotes standard deviation, Δ denotes C-index difference to DeepHit$^+$.

Algorithm	Mean C-index
FilterDeepHit$^+$ (ReliefF)	0.726
HybridDeepHit$^+$ (M)	0.725
HybridDeepHit$^+$ (P/Cut)	0.724
SparseDeepHit$^+$	0.724
HybridDeepHit$^+$ (M/Cut)	0.723
HybridDeepHit$^+$ (P)	0.721
RSF	0.719
FilterDeepHit$^+$ (SVM)	0.715
FilterDeepHit$^+$ (Anova)	0.713
AttentiveDeepHit$^+$	0.706
DeepHit$^+$	0.702
DeepHit	0.688
csCox	0.683
Fine-Gray	0.677

Table A.3: Average C-index performance of all models across all evaluation horizons and events on SEERsynth

Bibliography

Anaissi, A., P. J. Kennedy and M. Goyal (2011). 'Feature Selection of Imbalanced Gene Expression Microarray Data'. In: *2011 12th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*.

Andriole, G. L. et al. (2009). 'Mortality Results from a Randomized Prostate-Cancer Screening Trial'. In: *New England Journal of Medicine*.

Andriole, G. L. et al. (2012). 'Prostate Cancer Screening in the Randomized Prostate, Lung, Colorectal, and Ovarian Cancer Screening Trial: Mortality Results after 13 Years of Follow-up'. In: *JNCI: Journal of the National Cancer Institute*.

Bahdanau, D., K. Cho and Y. Bengio (2014). 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: *CoRR*.

Bauer, K. W., S. G. Alsing and K. A. Greene (2000). 'Feature screening using signal-to-noise ratios'. In: *Neurocomputing*.

Bellotti, T. and J. Crook (2009). 'Credit scoring with macroeconomic variables using survival analysis'. In: *Journal of the Operational Research Society*.

Belue, L. M. and K. W. Bauer (1995). 'Determining input features for multilayer perceptrons'. In: *Neurocomputing*.

Bergstra, J. and Y. Bengio (2012). 'Random Search for Hyper-parameter Optimization'. In: *J. Mach. Learn. Res.*

Bhattacharjee, S. et al. (2007). 'The Effect of Digital Sharing Technologies on Music Markets: A Survival Analysis of Albums on Ranking Charts'. In: *Management Science*.

Breiman, L. (2001). 'Random Forests'. In: *Machine Learning*.

Bron, E. E. et al. (2015). 'Feature Selection Based on the SVM Weight Vector for Classification of Dementia'. In: *IEEE Journal of Biomedical and Health Informatics*.

Chen, X. and H. Ishwaran (2012). 'Random Forests for Genomic Data Analysis'. In: *Genomics*.

Ching, T. et al. (2018). 'Opportunities and obstacles for deep learning in biology and medicine'. In: *Journal of The Royal Society Interface*.

Chorowski, J. et al. (2015). 'Attention-based Models for Speech Recognition'. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. MIT Press.

Collobert, R. and J. Weston (2008). 'A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning'. In: *Proceedings of the 25th International Conference on Machine Learning*. ACM.

Cox, D. R. (1972). 'Regression Models and Life-Tables'. In: *Journal of the Royal Statistical Society. Series B (Methodological)*.

Cox, D. R. and D. Oakes (1984). *Analysis of survival data*. Chapman and Hall.

Danacica, D.-E. and A.-G. Babucea (2010). 'Using Survival Analysis In Economics'. In: *Analele Stiintifice ale Universitatii "Alexandru Ioan Cuza" din Iasi - Stiinte Economice*.

DeGroot, M. H. and M. J. Schervish (2013). *Probability and statistics*. Pearson.

Eyal, A. et al. (2014). 'Survival Analysis of Automobile Components Using Mutually Exclusive Forests'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Faraggi, D. and R. Simon (1995). 'A neural network model for survival data'. In: *Statistics in Medicine*.

Ferri, F. et al. (1994). 'Comparative study of techniques for large-scale feature selection'. In: *Pattern Recognition in Practice IV*. North-Holland.

Fine, J. P. and R. J. Gray (1999). 'A Proportional Hazards Model for the Subdistribution of a Competing Risk'. In: *Journal of the American Statistical Association*.

Gerds, T. A. et al. (2013). 'Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring.' In: *Stat Med*.

Giot, P. and A. Schwienbacher (2007). 'IPOs, trade sales and liquidations: Modelling venture capital exits using survival analysis'. In: *Journal of Banking & Finance*.

Glorot, X. and Y. Bengio (2010). 'Understanding the difficulty of training deep feedforward neural networks'. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR.

Goh, G. B., N. O. Hodas and A. Vishnu (2017). 'Deep learning for computational chemistry'. In: *Journal of Computational Chemistry*.

Gohagan, J. K. et al. (2000). 'The Prostate, Lung, Colorectal and Ovarian (PLCO) Cancer Screening Trial of the National Cancer Institute: History, organization, and status'. In: *Controlled Clinical Trials*.

Goodfellow, I., Y. Bengio and A. Courville (2016). *Deep Learning*. MIT Press.

Gooley, T. A. et al. (1999). 'Estimation of failure probabilities in the presence of competing risks: new representations of old estimators.' In: *Stat Med*.

Gray, R. J. (1988). 'A Class of K-Sample Tests for Comparing the Cumulative Incidence of a Competing Risk'. In: *Ann. Statist.*

Gulati, R. et al. (2014). 'Individualized Estimates of Overdiagnosis in Screen-Detected Prostate Cancer'. In: *JNCI: Journal of the National Cancer Institute*.

Hall, M. A. and L. A. Smith (1999). 'Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper.' In: *FLAIRS conference*.

Hamon, J. (2013). 'Combinatorial optimization for variable selection in high dimensional regression: Application in animal genetic'. PhD thesis. Université des Sciences et Technologie de Lille - Lille I.

Harrell, F. E. J. et al. (1982). 'Evaluating the yield of medical tests.' In: *JAMA*.

He, K. et al. (2015). 'Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification'. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society.

Heijnsdijk, E. A. et al. (2012). 'Quality-of-Life Effects of Prostate-Specific Antigen Screening'. In: *New England Journal of Medicine*.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Houwelingen, H. van and H. Putter (2011). *Dynamic Prediction in Clinical Survival Analysis*. CRC Press, Inc.

Hsu, C.-N., H.-J. Huang and S. Dietrich (2002). 'The ANNIGMA-wrapper approach to fast feature selection for neural nets'. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*.

Ishwaran, H. et al. (2008). 'Random survival forests'. In: *Ann. Appl. Stat.*

Ishwaran, H. et al. (2014). 'Random survival forests for competing risks'. In: *Biostatistics*.

Jacobson, L. (2013). *Introduction to Artificial Neural Networks*. Accessed 20 August 2018. URL: `http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7`.

Jetley, S. et al. (2018). 'Learn to Pay Attention'. In: *International Conference on Learning Representations*.

Kabir, M. M., M. M. Islam and K. Murase (2010). 'A new wrapper feature selection approach using neural network'. In: *Neurocomputing*.

Kaplan, E. L. and P. Meier (1958). 'Nonparametric Estimation from Incomplete Observations'. In: *Journal of the American Statistical Association*.

Kartsonaki, C. (2016). 'Survival analysis'. In: *Diagnostic Histopathology*.

Katzman, J. et al. (2016). 'Deep Survival: A Deep Cox Proportional Hazards Network'. In:

Kavzoglu, T. and P. M. Mather (2002). 'The role of feature selection in artificial neural network applications'. In: *International Journal of Remote Sensing*.

Kingma, D. P. and J. Ba (2014). 'Adam: A Method for Stochastic Optimization'. In: *CoRR*.

Kira, K. and L. A. Rendell (1992). 'The Feature Selection Problem: Traditional Methods and a New Algorithm'. In: *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI Press.

Klein, J. P. and M. L. Moeschberger (2003). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer-Verlag.

Koller, D. and M. Sahami (1996). 'Toward Optimal Feature Selection'. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc.

Kononenko, I., E. Šimec and M. Robnik-Šikonja (1997). 'Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF'. In: *Applied Intelligence*.

Kordos, M. et al. (2017). 'Data selection for neural networks'. In: *Schedae Informaticae*.

Lawless, J. (1982). 'Statistical models and methods for lifetime data'. In: *Canadian Journal of Statistics*.

Lawless, J., J. Hu and J. Cao (1995). 'Methods for the estimation of failure distributions and rates from automobile warranty data'. In: *Lifetime Data Analysis*.

Lee, C. et al. (2018). 'DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks'. In: *AAAI*.

Leung, K.-M., R. M. Elashoff and A. A. Afifi (1997). 'Censoring Issues In Survival Analysis'. In: *Annual Review of Public Health*.

Li, Y., C.-Y. Chen and W. W. Wasserman (2016). 'Deep Feature Selection: Theory and Application to Identify Enhancers and Promoters.' In: *J Comput Biol*.

Luck, M. et al. (2017). 'Deep Learning for Patient-Specific Kidney Graft Survival Analysis'. In: *CoRR*.

Maas, A. L. (2013). 'Rectifier Nonlinearities Improve Neural Network Acoustic Models'. In:

Mares, M. A., S. Wang and Y. Guo (2016). 'Combining multiple feature selection methods and deep learning for high-dimensional data'. In: *Transactions on Machine Learning and Data Mining*.

Michalak, K. and H. Kwasnicka (2006). 'Correlation-based Feature Selection Strategy in Neural Classification'. In: *Sixth International Conference on Intelligent Systems Design and Applications*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Noone, A. et al. (2017). *SEER Cancer Statistics Review, 1975-2015, National Cancer Institute*. Based on November 2017 SEER data submission, posted to the SEER web site. Accessed 20 August 2018. URL: `https://seer.cancer.gov/csr/1975_2015/`.

Ooi, C. and P. Tan (2003). 'Genetic algorithms applied to multi-class prediction for the analysis of gene expression data'. In: *Bioinformatics*.

Pashayan, N., S. W. Duffy et al. (2015). 'Implications of polygenic risk-stratified screening for prostate cancer on overdiagnosis'. In: *Genetics In Medicine*.

Pashayan, N., P. D. Pharoah et al. (2015). 'Reducing overdiagnosis by polygenic risk-stratified screening: findings from the Finnish section of the ERSPC'. In: *British Journal Of Cancer*.

Pfirrmann, M. et al. (2015). 'Prognosis of long-term survival considering disease-specific death in patients with chronic myeloid leukemia'. In: *Leukemia*.

Pinsky, P. F. et al. (2017). 'Extended mortality results for prostate cancer screening in the PLCO trial with median follow-up of 15 years'. In: *Cancer*.

Prorok, P. C. et al. (2000). 'Design of the Prostate, Lung, Colorectal and Ovarian (PLCO) Cancer Screening Trial.' In: *Control Clin Trials*.

Prostate cancer mortality statistics, Cancer Research UK (2018). Accessed 20 August 2018. URL: `http://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/prostate-cancer/mortality`.

Rezaianzadeh, A. et al. (2009). 'Survival analysis of 1148 women diagnosed with breast cancer in Southern Iran'. In: *BMC Cancer*.

Rosenblatt, F. (1958). 'The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain'. In: *Psychological Review*.

Roy, D., K. S. R. Murty and C. K. Mohan (2015). 'Feature selection using Deep Neural Networks'. In: *2015 International Joint Conference on Neural Networks (IJCNN)*.

Ruangkanokmas, P., T. Achalakul and K. Akkarajitsakul (2016). 'Deep Belief Networks with Feature Selection for Sentiment Classification'. In: *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). 'Learning representations by back-propagating errors'. In: *Nature*.

Saeys, Y., I. Inza and P. Larrañaga (2007). 'A review of feature selection techniques in bioinformatics'. In: *Bioinformatics*.

Schröder, F. H. et al. (2009). 'Screening and Prostate-Cancer Mortality in a Randomized European Study'. In: *New England Journal of Medicine*.

Schröder, F. H. et al. (2012). 'Prostate-Cancer Mortality at 11 Years of Follow-up'. In: *New England Journal of Medicine*.

Schröder, F. H. et al. (2014). 'Screening and prostate cancer mortality: results of the European Randomised Study of Screening for Prostate Cancer (ERSPC) at 13 years of follow-up'. In: *The Lancet*.

Setiono, R. and H. Liu (1997). 'Neural-network feature selector'. In: *IEEE Transactions on Neural Networks*.

Siedelecki, W. and J. Sklansky (1988). 'On Automatic Feature Selection'. In: *International Journal of Pattern Recognition and Artificial Intelligence*.

Silver, D. et al. (2017). 'Mastering the game of Go without human knowledge'. In: *Nature*.

Sivagaminathan, R. K. and S. Ramakrishnan (2007). 'A hybrid approach for feature subset selection using neural networks and ant colony optimization'. In: *Expert Systems with Applications*.

Skalak, D. B. (1994). 'Prototype and feature selection by sampling and random mutation hill climbing algorithms'. In: *Machine Learning Proceedings 1994*. Elsevier.

Srivastava, N. et al. (2014). 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. In: *Journal of Machine Learning Research*.

Thompson, I. M. et al. (2006). 'Assessing prostate cancer risk: results from the Prostate Cancer Prevention Trial.' In: *J Natl Cancer Inst*.

Trunk, G. V. (1979). 'A Problem of Dimensionality: A Simple Example'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Tsai, C.-F. and Y.-C. Hsiao (2010). 'Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches'. In: *Decision Support Systems*.

Vaswani, A. et al. (2017). 'Attention is All you Need'. In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc.

Vickers, A. J. et al. (2013). 'Strategy for detection of prostate cancer based on relation between prostate specific antigen at age 40-55 and long term risk of metastasis: case-control study'. In: *BMJ*.

Yen, A. M.-F. et al. (2015). 'Prostate cancer screening using risk stratification based on a multi-state model of genetic variants.' In: *Prostate*.

Yu, L. and H. Liu (2003). 'Feature selection for high-dimensional data: A fast correlation-based filter solution'. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*.

Zhu, X., J. Yao and J. Huang (2016). 'Deep convolutional neural network for survival analysis with pathological images'. In: *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*.