Imperial College London

Imperial College London
Derpartment of Mathematics

Joint analysis of clinical risk factors and 4D cardiac motion for survival prediction using a hybrid deep-learning network

Shihao Jin

CID: 01570974

Supervised by Professor Axel Gandy and Dr Declan O'Regan

16th September 2019

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Date:

Abstract

In recent years, there has been an increasing interest in applying deep-learning (DL) techniques to medical research, due to the arising of more and more structurized and complex data - clinical and genomic for instance. With its flexibility and ability for handing large data set, deep-learning network shows many advantages over traditional statistical models, especially for prediction task. Particularly, in survival analysis area, many traditional topics, including time-dependent covariates, competing risks, begin to be explored in deep-learning framework. In this thesis, two general extensions are proposed based on Bello et al. (2019), in which they predicted the survival outcomes using high dimensional cardiac motion data for a cohort of patients with pulmonary hypertension.

In the first extension, a novel approach is come up for joining the high dimensional time-resolved cardiac motion features and comparably low dimensional clinical factors to increase the survival prediction performance. Different methods are also suggested to find the the optimal way to insert these extra covariate factors in deep networks. Especially, correlation analysis between autoencoder latent code and covariates features are conducted for explaining the level of association between both and exploring how these two streams of information interact. This extension opens up new directions for the use of heterogeneous clinical factors to push artificial intelligence towards personalized medical treatment. It is believed that similar approaches could also be used to introduce knowledge of genetic variants to such survival networks to improve outcome prediction by jointly analysing cardiac motion traits with inheritable risk factors.

The other extension lies in relaxing the Cox proportional hazard model and instead, harnessing the flexibility of the network to learn the probability distribution of survival time directly (Lee et al., 2018). It is also capable of modelling time-varying relationship between covariates and the risk. The novelty of this part is that, to my best knowledge, it is the first time to implement probability distribution prediction with cardiac motion data set mixed with clinical factors, and achieves better performance over the optimal model described in the first extension. The significance of this work is providing a new heuristic thinking for survival models in deep-learning framework, forcing the network to learn different elements in survival analysis.

Acknowledgements

I would first like to thank my supervisors Professor Gandy for his expert advice and encouragement, as well as Dr O'Regan for providing resources in the lab. Every Wednesday afternoon's discussion with Professor Gandy really inspired me to walk further on the project.

I would also like to thank Dr. Savioli, who gave me lots of help on implementation and pushed me to complete a paper for a NIPS workshop.

The appreciation would be finally given to my parents for their sponsorship for the MSc programme and Joyce for her brilliant suggestions in thesis writing.

Table of Contents

1. Introduction

Heart failure is a class of disease influencing twenty-six million people worldwide and increasing in prevalence (Savarese and Lund, 2017). It is also a disease with high mortality where the choice of treatment depends on individual risk stratification (Galiè et al., 2015). In particular, this thesis targets at pulmonary hypertension, characterized by right ventricular (RV) dysfunction.

Pulmonary hypertension is high blood pressure in the blood vessels that supply the lungs (pulmonary arteries), defined by NHS (2017). It affects the arteries in the lungs and the right side of the heart. To be specific, the tiny arteries in the lungs, called pulmonary arterioles, and capillaries firstly become narrowed, blocked or destroyed. This makes it harder for blood to flow through the lungs and raises pressure within the lungs' arteries. As the pressure builds up, the heart's lower right chamber (right ventricle) must work harder to pump blood through the lungs, eventually causing the heart to weaken and fail (Simon and Pinsky, 2011).

Though Pulmonary hypertension is a rare condition that can affect people of all ages, the mortality rate is quite high once diagnosed. The survival of patients used to be around three to five years, and it could take two or more years to correctly diagnose (Flavell, 2018). Luckily, since the disease was first identified in 1891, the life expectancy of patients with pulmonary hypertension has been deemed an increase, especially for the development of specialized therapies for pulmonary hypertension (Sitbon et al., 2014), beginning with the introduction of Flolan in 1995. Nevertheless, it is pointed out by a recent research that the mortality rate of patients with pulmonary hypertension is still more than three times higher than in the control group of healthy people (Chang et al., 2016).

With the development of high resolution imaging techniques, cardiac phenotype offers helpful information and has been widely used for disease diagnosis and prognosis (Ahmad et al., 2014). The motivation of this thesis is to predict patients' risks of death based on cardiac motion data, as well as clinical factors, so that doctors can give more personalized therapy. This type of study is called survival analysis and the response of interest is the time until the occurrence of some event (Kalbfleisch and Prentice, 2011). Such event is also called a failure, hence survival data is referred to as time-to-event data or failure time data, as well. For the purpose of incorporating covariates to explain the survival outcomes, methods are extensively studied for time-to-event data. Section 2.1, following the evolution of Cox proportional hazard model (Cox, 1972), reviews several model extensions and relevant inference techniques. Section 2.2.1 and 2.2 introduce the basic concepts in neural networks and present more recent literatures on survival analysis with neural network framework, according to the loss functions they use.

Due to the structure and complexity of cardiac motion data, this thesis dives into the

state-of-the-art neural network framework and focuses on:

- chapter 4: how to incorporate (asymmetric) extra information in survival models involving deep-learning network and how to understand the interaction between these two streams of information. The asymmetry is in the sense that the new covariates are in much lower dimension compared with that of existed data.
- chapter 5: how to relax the restriction in the Cox model and modify the survival part in the neural network to improve the failure risk prediction.

The rest part of the thesis is organized in the following: chapter 3 is for introduction to data pre-processing and data formatting, and chapter 6 summarizes the overall thesis and points out some potential further works.

2. Review on survival analysis

Survival analysis is aimed to extract information from time-to-event data, i.e. the time up to the occurrence of an event of interest. It is widely used in epidemiological studies and econometric researches, where the incomplete observation of failure times frequently arises due to censoring. There are three types of censoring:

- right-censoring, e.g. the event of interest occurs after the observation ends,
- left-censoring, e.g. the event occurs before the observation starts,
- interval-censoring, e.g. researchers check the status of the subjects periodically and what they only know is that the event occurs within an interval.

Among them, right-censoring is the most common one and the censoring always points right-censoring without specification in the thesis. Strictly, survival data can be organized as followed. Assume that the underlying survival(event or failure) time and censoring time are two nonnegative random variables, X and C, respectively. What can be observed is $T = \min(X, C)$ and a censoring indicator $\Delta = I(X < C)$, but the goal is to infer X based on the observations $\{t_i, \delta_i\}_{i=1}^{N}$ of (T, Δ), where N is the number of samples. An important assumption throughout the thesis is that X and C are statistically independent.

There are five kinds of functions to characterize the distribution of real survival time X (assumed to be continuous here for simplicity; it can be discrete or mixed; the failure refers to mortality in the following introduction).

- Cumulative distribution function, $F(x) = P(X \leq x)$, representing the probability that an individual dies before or at time x. $F(x)$ must be a monotone non-decreasing function and $F(0) = 0$, $F(\infty) = 1$.
- Survival function, $S(x) = P(X > x) = 1 - F(x)$, meaning the probability that an individual survives over time x. Consequently, $S(x)$ is a monotone non-increasing function and $S(0) = 1$, $S(\infty) = 0$.
- Probability density function, $f(x)$, is defined as an individual's unconditional instant mortality rate, $f(x) = \lim_{\Delta x \to 0} \frac{P(x < X \leq x + \Delta x)}{\Delta x}$. As the presumed continuity of X, $F^{'}(x) = f(x)$.
- Hazard function, $\lambda(x) = \lim_{\Delta x \to 0} \frac{P(x < X \leq x + \Delta x) | X > x)}{\Delta x}$, refers to the instant mortality rate at time x, conditioning on surviving over time x. After some rearrangement, $\lambda(x) = \frac{f(x)}{S(x)}$.

- Cumulative hazard function, defined as $\Lambda(x) = \int_0^x \lambda(t)dt = -ln(S(x))$, shows the accumulated mortality risk from time 0 to x. The relationship between the cumulative hazard function and the hazard function is similar to that between the cumulative distribution function and the probability density function.

As shown above, the rest four can be deduced from arbitrary one of the five. For ease of notation, all of the above functions appearing in the following are for real survival time X, not for observed T.

This chapter overviews the survival analysis mainly from the perspective of regression models. Traditional models and state-of-the-art neural networks are separately reviewed in section 2.1 and 2.2, while their interactions are discussed in section 2.3. As an widely used nonparametric estimate for survival distribution, Kaplan-Meier estimator is mentioned in section 2.1 as well.

2.1. Traditional survival analysis

Kaplan-Meier estimate (Kaplan and Meier, 1958) was proposed very early and is still popular in survival analysis. Under the assumption that individuals will only be at risk at the death time, Kaplan-Meier estimator is able to model very flexible survival functions. Moreover, it is in non-parametric paradigm and there is no requirement for specification of the underlying distribution.

Nevertheless, it suffers the disadvantage of not incorporating subjects' covariates, hence being restricted by presuming a homogeneous population. Once additional information about the subjects is given, Cox proportional hazard model (Cox, 1972) provides a regression method to accommodating explanatory variables. The art of Cox's model lies that it regards the hazard as a semi-parametric function of a $p \times 1$ vector of covariates $\boldsymbol{Z}$:

$$\lambda(t; \boldsymbol{Z}) = \lambda_0(t)\exp(\boldsymbol{Z}^T\boldsymbol{\beta}). \tag{2.1}$$

where $\lambda_0(t)$ is an unspecified baseline hazard function and $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown regression parameters. The name, "proportional hazard", is originated from the view that the hazard ratio of two given subjects with covariates $\boldsymbol{Z}$ and $\boldsymbol{Z}'$ is in constant proportion at all time. To estimate $\boldsymbol{\beta}$ given the data $\{t_i, \delta_i, \boldsymbol{Z}_i\}_{i=1}^N$, following the idea of maximising likelihood, Cox came up with partial likelihood method under certain assumptions (Cox, 1972):

> " In the absence of knowledge about λ_0, the time intervals between deaths can provide little or no information about $\boldsymbol{\beta}$, as their distribution will depend heavily on λ_0."

Therefore, it is only the event time ordering of the subjects that holds information about $\boldsymbol{\beta}$. The contribution to the likelihood function of $\boldsymbol{\beta}$ for an individual, with covariates $\boldsymbol{Z}_i$

and dead at time t_i, is

$$\begin{aligned} & P(\text{individual } i \text{ with covariates } \boldsymbol{Z}_i \text{ dies at } t_i | \text{ arbitrary one death at } t_i) \\ = & \frac{P(\text{individual } i \text{ with covariates } \boldsymbol{Z}_i \text{ dies at } t_i)}{P(\text{arbitrary one death at } t_i)} \\ = & \frac{\lambda(t_i; \boldsymbol{Z}_i)}{\sum_{j \in R_{t_i}} \lambda(t_i; \boldsymbol{Z}_j)} \\ = & \frac{\exp(\boldsymbol{Z}_i^T \boldsymbol{\beta})}{\sum_{j \in R_{t_i}} \exp(\boldsymbol{Z}_j^T \boldsymbol{\beta})}, \end{aligned} \tag{2.2}$$

where N represents the number of subjects and t_i is ith subject's observed survival or censoring time. And R_{t_i} is an index set that $\forall j \in R_{t_i}$, jth subject is alive or censored. The above formula is called partial likelihood, in the sense that it just uses the partial information as the baseline hazard is not specified. Maximising the partial likelihood can get an estimate of $\boldsymbol{\beta}$. The formula (2.2) is also important in constructing loss function for training neural networks, details of which will be described in section 4.1.1.

Due to its simplicity and versatility, Cox proportional hazard model leads the literatures on regression models in survival analysis, while its shortcomings are obviously. De Laurentiis and Ravdin (1994) pointed out three of them:

- The proportional hazards assumption does not hold,
- The variables have a complex and unknown relationship with the outcome, i.e. not just a semi-parametric form linked with a exponential function,
- There are interactions between the variables, i.e. not just a linear combination.

There are plenty of generalized models were come up with in the subsequent decades after Cox's papers in 1972 and 1975. With longitudinal data, Cox's model with time-dependent covariates has been studied by many authors, including Andersen and Gill (1982), Lin and Wei (1989). Generally, they regarded the process of event occurrence as a counting process and explored the asymptotic properties with martingale techniques. Another extension of Cox model diverts to incorporate time-varying coefficients to evaluate the covariates' effects changing with time. Zucker et al. (1990), Cai and Sun (2003) and Tian et al. (2005), were examples, where the first one used roughness penalization for coefficient functions and the latter two employed kernel smoothing techniques. A more general model was proposed by Fan et al. (2006),

$$\lambda(t; \boldsymbol{Z}(t), W(t)) = \lambda_0(t) \exp(\boldsymbol{Z}(t)^T \boldsymbol{\beta}(W(t)) + g(W(t))), \tag{2.3}$$

where $\boldsymbol{\beta}(\cdot)$ and $g(\cdot)$ are unknown coefficient functions, describing the extent to which the relationship varies with the level of the exposure variable $W(t)$. When the variable $W(t)$ is time, rather than a covariate variable, model (2.3) becomes a time-dependent coefficient model mentioned above. In Fan et al. (2006), they assume certain smoothness of $\boldsymbol{\beta}(\cdot)$ and $g(\cdot)$, and also employ local linear techniques to estimate the coefficient functions. What's more, a penalized (smoothly clipped absolute deviation penalty, SCAD) local likelihood estimator was proposed to select important risk factors in the model.

Another research line about survival regression models follows addictive nonparametric model (Aalen, 1989), in which the hazard function is directly a linear combination of time-varying coefficients and time-dependent covariates, so-called Aalen's additive function. Although this model is unconventional in the sense that it can stray into negative values for the hazard rate, a more correct description of the actual relationship than a multiplicative model is shown in some cases (Timmreck, 2002), and the linearity assigns it simpler theoretical estimate. Later, a natural idea is to combine the Cox's model and the additive model to cross-fertilize the model fitting and prediction. The idea was firstly proposed by Scheike (2002) and abundant subsequent work in this direction has been carried out by Martinussen and Scheike (2007). Particularly, the model replaces the unspecified baseline function in Cox's model by an Aalen's additive function:

$$\lambda(t; \boldsymbol{Z}(t), \boldsymbol{X}(t)) = \boldsymbol{X}(t)^T \boldsymbol{\alpha}(t) \exp(\boldsymbol{Z}(t)^T \boldsymbol{\beta}(t)), \tag{2.4}$$

where there are different characteristics $\boldsymbol{Z}(t), \boldsymbol{X}(t)$ for baseline and exponential term, respectively, and $\boldsymbol{\alpha}(t)$, $\boldsymbol{\beta}(t)$ are unknown coefficient functions. In the meanwhile, the parametric form for baseline also eases the estimation compared with that in Cox's model.

The final model worthy of mentioning is accelerate failure time (AFT) model, which is in the form of

$$\lambda(t; \boldsymbol{Z}) = \lambda_0(t \cdot \exp(\boldsymbol{Z}^T \boldsymbol{\beta})) \exp(\boldsymbol{Z}^T \boldsymbol{\beta}). \tag{2.5}$$

The significance of the model lies in the covariates acting multiplicatively on time so that their effect will accelerate or decelerate time to failure for one individual relative to another. Statistical inference for accelerate failure time model was summarized in Kalbfleisch and Prentice (2011).

2.2. Survival analysis with neural networks

In recent years, as a type of more flexible but complex model, artificial neural networks, especially deep networks, are substantially used in survival modelling and prediction. In the following, an introduction to neural networks are given in section 2.2.1 and in section 2.2.2, different networks are discussed according to different loss functions they used, reflecting their underlying statistical ideas.

2.2.1. An introduction to neural networks

As a relatively recent evolutionary machine learning algorithm, artificial neural network (ANN) is capable of modelling complex non-linear relationship between input X and the target value K. The standard structure (figure 2.1) of the ANN consists of three components, the input (i.e. predictor variables) $X = (X_1,\ X_2,\ ..., X_J)$, a hidden layer $H = (H_1,\ H_2,\ ..., H_H)$ and the output target values $K = (K_1,\ K_2,\ ..., K_K)$. In each layer, there can be one or more nodes (also called neurons), and usually they are fully connected between adjacent layers. This connections have unique directions, in the sense of from the input to hidden layer, from hidden layer to the output. It is so-called feed forward neural network, which is the most common one and all mentioned neural networks in the thesis are of this type. On each connection, a weight is assigned, and these

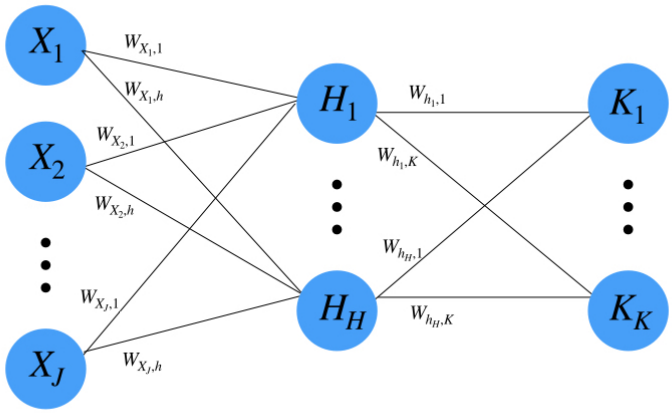


Figure 2.1.: Architecture of the single hidden layer neural network. X represents the inputs, H the hidden nodes, and K the output nodes. The nodes are unidirectionally connected with the corresponding weights W.

weights are the parameters of the neural network.

Suppose the network is trained on a data set of N subjects and the network will estimate a value $\hat{y}_k(\boldsymbol{x}_i, \boldsymbol{w})$ for ith individual at output node K_k, $k = 1, 2, \ldots, K$, given the input $\boldsymbol{x}_i$ and all the weights $\boldsymbol{w}$. Mathematically, a feed forward neural network can be represented as followed,

$$\hat{y}_k(\boldsymbol{x}_i, \boldsymbol{w}) = f_k(\alpha_k + \sum_{h=1}^{H} w_{hk} f_h(\alpha_h + \sum_{j=1}^{J} w_{jh} x_{ij})), \tag{2.6}$$

where α_k and α_h are the corresponding intercepts, f_k and f_h are the activation functions, and x_{ij} is the jth component of $\boldsymbol{x}_i$. Activation functions are used to mimic the behavious of neurons - only the input information is strong enough so that the neurons can be activated and pass the information to the next layer. There are multiple choices for activation functions and four of them are listed below:

- linear activation function: $f(x) = x$. It is rarely used, except for some output layers to preserve certain practical meanings.
- sigmoid (logistic) activation function: $f(x) = \dfrac{1}{1+e^{-x}}$. It is like a step function but has a smooth gradient, which is helpful in training process.
- softmax activation function: $f(x_1, x_2, \ldots, x_p) = (\dfrac{e^{x_1}}{\sum_{j=1}^{p} e^{x_j}}, \dfrac{e^{x_2}}{\sum_{j=1}^{p} e^{x_j}}, \ldots \dfrac{e^{x_p}}{\sum_{j=1}^{p} e^{x_j}})$. So this is a multivariate map $f : \mathbb{R}^p \mapsto \mathbb{R}^p$. Note that all the elements in the output vector sum up to 1, indicating softmax activation function is particularly suitable for the layer representing probability.
- ReLU activation function: $f(x) = \max\{0, x\}$. This activation function is widely used as it accelerates the convergence of the training process. Also, ReLU will

encourage sparsity of the networks, resulting in concise models that often have better predictive power and less overfitting.

The network is trained until the defined loss, L, is minimized. The loss is a function of $\hat{y}_k(\boldsymbol{x}_i, \boldsymbol{w})$ and the observed values y_{ik}^o, $i = 1, 2, \ldots, N$, $k = 1, 2, \ldots, K$, where the superscript o denotes an observation. For different problems, discrepant loss functions are applied. Details will be introduced in the following chapters. The minimization of the loss function is conducted through back-propagation algorithm, which was firstly brought up by Bryson and Ho (1969).

If multiple hidden layers are put in the neural networks, it evolves to the deep-learning network, which is believed to have more powerful ability to learn the input information and own better validation performance (Brahma et al., 2015). Along with that, the training budget increases with the deepening of neural networks (traditional gradient descent-based algorithms, like back-propagation, are not applicable any more) and overfitting issue arises. For the former, mini-batch training is proposed, in which the algorithm evenly divides the whole data set into several batches and calculates the loss for each batch separately, but only updates the model parameters, e.g. weights, after all the batches have been evaluated. In this way, not only the faster convergence is achieved, but also it helps to save the computing memory and encourages parallel processing. If the batch size is exactly 1, the algorithm is called Stochastic Gradient Descent (Bottou, 2010). For the latter problem, dropout and penalization are introduced. Dropout is simple, just randomly setting some of the nodes' value to 0. As for the penalization, it adds a penalty term, $p(\cdot)$, to the weights in the loss function:

$$L^\star = L + \lambda p(\boldsymbol{w}), \tag{2.7}$$

where λ is a penalization parameter. Usually, the penalty function, $p(\cdot)$, is l_1 norm to push some weights to 0. These two techniques promote the robustness of the model and consequently avoid or relieve the overfitting.

2.2.2. An overview of existed methods

A large quantity of the literatures, especially before 2010, gave preference to relative entropy or cross entropy types of loss function (regarding survival analysis as a classification problem), and varied on how to label the target. Some authors suggested to label survival probability, by discretizing the time line into K intervals, i.e. $I_1, \ldots, l_K$. Street (1998) was an old example, in which they proposed a simple artificial neural network with one hidden layer, and the output was the discretized survival function, e.g. the first unit represented the failure occurring after l_1. The relative entropy error function was used, in order to maintain the interpretation of the outputs as probabilities:

$$L = \sum_{n=1}^{N} \sum_{k=1}^{K} \Big[\frac{1}{2}(1 + T_n^k) \log \frac{1 + T_n^k}{1 + O_n^k} + \frac{1}{2}(1 - T_n^k) \log \frac{1 - T_n^k}{1 - O_n^k} \Big], \tag{2.8}$$

where N is the number of subjects, T_n^k is the nth subject's target value for output node k and O_n^k is the corresponding output value. For this kind of loss function, two labelling examples are shown in table 2.1. It is understandable to note that for an uncensored case, the target vector is assigned 1s up until the interval of event's

occurrence, and -1s afterwards. As for a censored case, the corresponding survival probability can be labelled iteratively: $S_t = 1, t$ is before or at censoring interval; $S_t = S_{t-1}(1 - \hat{\mu}_t), t$ is after censoring interval, where $\hat{\mu}_t$ is the estimated hazard by Kaplan-Meier maximum likelihood approximation introduced by equation A.2 in appendix A.1. In a follow-up work (Chi et al., 2007), a slight improvement was come up with in the definition of the target vectors and the activation function used - sigmoid instead of hyperbolic tangent.

	K_1	K_2	K_3	K_4	K_5	K_6
Uncensored Patient died at K_3	1	1	1	-1	-1	-1
Censored patient censored at K_4	1	1	1	1	0.8	0.5

Table 2.1.: Two labelling examples for Street (1998)'s neural network. 0.8 and 0.5 are two proper numbers for example. In practice, they should be derived iteratively described in the above text.

Nilsaz-Dezfouli et al. (2017) proposed another method based on cross-entropy loss. They turned to consider the cumulative distribution function. Rather than using a network to produce multiple output, separate networks were constructed and each of them was designed to output the probability of dying at or before one discretized time slot. For a given network for interval $k, k = 1, \ldots, K$, an individual experiencing the event in the interval k would be assigned a 1 as target; if a patient gets censored after that interval, the output would target 0; and for a patient censored before or within the interval, the target $\hat{p}$, was estimated using the Cox's model (mentioned in section 2.1). With respect to kth network, the cross-entropy function will be minimized:

$$L_k = -\sum_{n=1}^{N} \Big[T_n^k \log O_n^k + (1 - T^k) \log(1 - O_n^k) \Big], \tag{2.9}$$

following the notation in the previous paragraph.

These two methods work in most cases, while a serve issue will be met, because there is no guarantee that the estimated survival function and cumulative distribution function are monotone. Something weird may happen, e.g. the estimated probability of dying after 3th interval may be smaller than that of 5th interval. This is the partial reason why some related works tried to focus on the hazard function. Biganzoli et al. (1998) described a partial logistic artificial neural network approach. A single output was defined in the neural network for the prediction of ith patient's hazard at kth interval, $\hat{\mu}_i(l_k)$. The target matrix is $\boldsymbol{d}$, with $d_{ik} = 1$ if ith patient gets censored at interval l_k and $d_{ik} = 0$, otherwise. Now the overall cross-entropy error function is :

$$L = -\sum_{i=1}^{N} \sum_{k=1}^{K} \Big[d_{ik} \log \hat{\mu}_i(l_k) + (1 - d_{ik}) \log(1 - \hat{\mu}_i(l_k)) \Big], \tag{2.10}$$

Note that in this situation, the cross-entropy function is equivalent to the negative logarithm of the likelihood $L = \prod_{i=1}^{N} \prod_{k=1}^{K} \hat{\mu}_i(l_k)^{d_{ik}} (1 - \hat{\mu}_i(l_k))^{1-d_{ik}}$. Biganzoli et al. (1998)'s model is a key development in survival analysis with neural networks, in the sense that

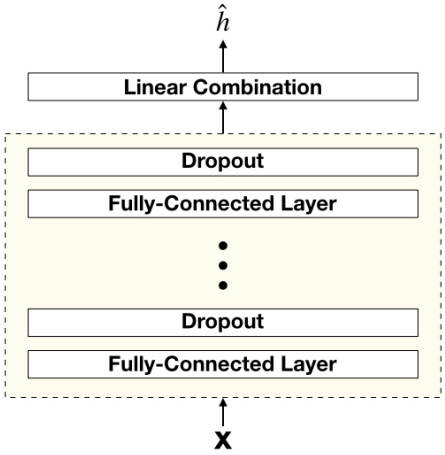


Figure 2.2.: The architecture of DeepSurv. $\boldsymbol{x}$ is the input data to the network. $\hat{h}$ is the predicted relative risk.

it is the first time to unify classification paradigm and likelihood thinking, and boosts the following models, like Lee et al. (2018)'s DeepHit model, which will be reviewed later.

Another extensively used loss is Cox type loss. The main characteristic of this kind of network lies in the single linearly activated output, which stands for the predicted relative risk $\hat{h} = \boldsymbol{Z}^T\boldsymbol{\beta}$ in equation 2.1. Consequently, the loss will be defined as negative log-partial likelihood referring to equation 2.2. Faraggi and Simon (1995) were the first ones who proposed a feed-forward neural network, which required no prior assumption of linear combination of the input but based on Cox proportional hazard model. This model is a standard three-layer artificial neural network, like the three previous models, while sometimes it has not been shown to outperform the linear Cox proportional hazard model, partly due to the limited depth of the network.

Katzman et al. (2018) had the similar idea with Faraggi and Simon (1995)'s, but a more deep structure. The novel network, DeepSurv, propagates the inputs through a number of hidden layers, consisting of fully-connected nonlinear activation functions, followed by dropout. The final layer, as usual, is a single unit that performs a linear combination of the hidden features. As we can see in figure 2.2, DeepSurv provides a deep non-linear version of the Cox's model with a further powerful learning capability to perceive the connection among the patients' characteristics and the survival consequences. Almost at the same time as Katzman et al., Ching et al. (2018) applied a similar network, Cox-nnet, to a set of high-throughput omics data. They also employed regularization techniques, like ridge penalty and dropout for the sparsity of the model. They found the network with one hidden layer got the best prediction performance for transcriptomics data. Later, Bello et al. (2019) conducted a hybrid deep-learning model to understand the behaviour of moving cardiac in image sequences for survival prediction. The distinction was that they firstly applied a denoising autoencoder for dimension reduction and then fed the compressed low dimensional latent code to a Cox's model. They claimed that this network achieved outperformance of prediction accuracy compared with that

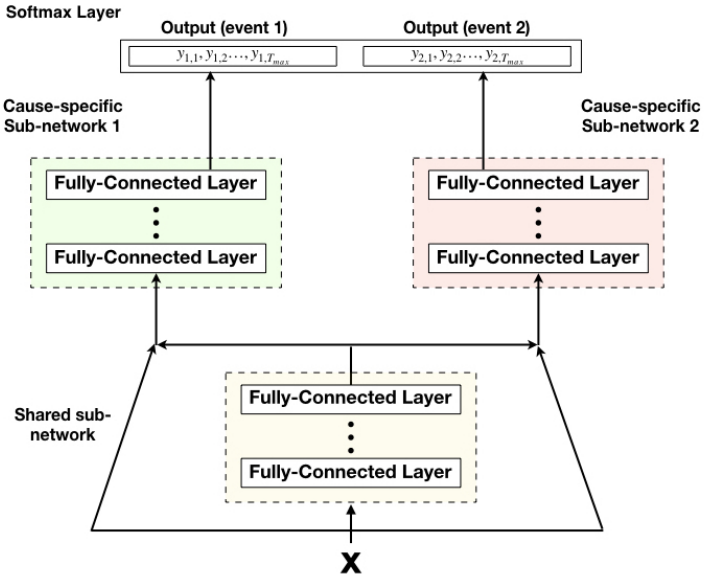


Figure 2.3.: The architecture of DeepHit with two competing events. The output $(y_{1,1}, y_{1,2}, \ldots, y_{1,T_{\max}}, y_{2,1}, y_{2,2}, \ldots, y_{2,T_{\max}})$ stands for the probability mass for the joint distribution.

of human benchmark. The details of the model will be further described in section 4.1.1.

After a large number of application of Cox type loss, it is realized that the presuming concrete forms of the link between hazard function and covariates is too constricted. Lee et al. (2018) proposed an idea to harness the flexibility of network to learn the probability density of survival time directly. Using the similar discretization trick, the network, Deephit, tries to estimate the probability mass for each interval, e.g. the first unit in the output layer represents the probability of dying in the first interval. And the induced loss function is the negative log-empirical likelihood which will be detailedly introduced in section 5.1. The other novelty of this network lies in its capacity of handling competing risks. In the example of two competing events (figure 2.3), the input data $\boldsymbol{x}$ is firstly fed to a fully-connected shared sub-network, the output of which is then sent to two cause-specific sub-networks. The model also maintains a residual connection (He et al., 2015) from the input covariates into the input of each cause-specific sub-networks. The products of the two sub-networks are finally linked to a softmax layer comprising the joint distribution of the two competing events. In their following work (Lee et al., 2019), DeepHit was evolved to deal with time-dependent covariates and an embedded imputation procedure was designed for missing values. In their experiment, Deephit behaved significantly better prediction over other state-of-the-art methods, e.g. DeepSurv above and random survival forest below.

Remark 1. Competing risks problem arises in the case when an individual can experience one of any m failure types, $m \geq 2$. For instance, one person may die from a disease or just by accident. These are two types of risks, but people can only experience one of them in their lives.

Random survival forest (Ishwaran et al., 2008) is not strictly a neural network, but also widely used and gives pretty good performance. It is a tree-based method and for each tree in the forest, samples are boostrapped; at each node in a tree, features are boostrapped. Feature that maximizes the log-rank statistic (equation A.8 in Appendix) is selected to split the nodes. The cumulative hazard function (CHF, mentioned at the beginning of the chapter) is estimated at the leaf nodes, and an subject's CHF is computed via averaging over all the trees in the ensemble.

2.3. Discussion

In the above literature review, it is found that traditional survival analysis mainly focuses on modelling hazard function. This is reasonable, because hazard function is more flexible compared with cumulative distribution function, survival function and probability density function, in the sense that any non-negative functions defined on $(0, \infty)$ can be hazard functions. Normally in theory, asymptotic properties will be firstly derived on the hazard function, then deduced to other functions. In neural networks, the hazard function also enjoys benefit. In Biganzoli et al. (1998)'s model, outputting of hazard helps to establish the bridge between cross-entropy function and the likelihood. Moreover, there is no concern about the shape of the estimated function (the activation function ensures the positivity of the output values).

The evolution of the traditional survival models is aimed to incorporate more parameters and makes the model more close to the reality. Neural networks finally achieve the goal. The hidden layers and the activation functions are capable of characterizing the non-linearity and the interaction among input. Nevertheless, neural networks are more like a black box, hard to interpret what really happens in the training process. Hence, for some inference problems, like evaluating the effect of certain covariates, traditional models are never out of date, while neural networks dominate for prediction tasks. In the thesis, massive effort is given to understand the behaviours of the neural networks, as discussed in section 4.4 and 5.5.

3. Data description and pre-processing

This chapter describes the data used in application throughout the rest of the thesis, including high dimensional heart moving trajectories data and conventional clinical factors. The acquisition for the former is detailed in section 3.1 and descriptive statistical analysis for the latter part is shown in section 3.2. The last section briefly introduce the data pre-processing, which helps for the convergence of the neural networks. Except for survival data, cardiac motion data and clinical factors are assumed to be measured at the time origin.

3.1. Cardiac image sequences

The data are collected from patients registered at the National Pulmonary Hypertension Service at the Imperial College Healthcare NHS Trust between May 2004 and October 2017. Criteria for inclusion are the documented diagnosis of Group 2, 3 and 4 pulmonary hypertension investigated by right heart catheterization with a mean pulmonary artery pressure $\geq$ 25mmHg and pulmonary capillary wedge pressure $<$ 15mmHg; and signs of chronic thrombo-embolic disease presented on either ventilation-perfusion scintigraphy or computed tomography pulmonary angiography (Gopalan et al., 2017). The number of included patients is 302 and 28% (85 of 302) are observed to the death.

The hyper-high dimensional image data are derived from cardiac magnetic resonance (CMR), which photographs the heart in any anatomical plane for dynamic assessment of the functions. The heart is scanned in 9 slices across 20 temporal phases and ventricles are automatic segmented in each slice (see an example in figure 3.1). For the prediction task, patients diagnosed with pulmonary hypertension are characterized by right ventricular (RV) dysfunction, so it is reasonable to focus on right ventricular and construct its 3D mesh based on segmented slice images for each temporal phase. Figure 3.2 is a visualization of a patient's right ventricular at a specific temporal phase.

In this way, the series of 20-temporal-phase 3D meshes can be used to characterize the cardiac motion. For the purpose of prediction, a sparser version of the meshes with 202 vertices is utilized by down-sampling with a factor of 0.1. Anatomical correspondence is preserved in this process in the way of extracting the same vertices across all meshes.

Finally, some notations are introduced for generating $\boldsymbol{x}$, an extremely long vector that describes the RV motion. Let (a_{vs}, b_{vs}, c_{vs}) represent the Cartesian coordinates of the vertex $v (v = 1, \ldots, 202)$ at the sth time frame $(s = 1, ..., 20)$ of the cardiac cycle. At each time frame $s = 2, 3, ..., 20$, the coordinate-wise displacement of each vertex from its position at time frame 1 is computed. This yields $\boldsymbol{x}$,

$$\boldsymbol{x} = (a_{vs} - a_{v1}, b_{vs} - b_{v1}, c_{vs} - c_{v1})_{2 \leq s \leq 20}^{1 \leq v \leq 202}, \tag{3.1}$$

which has length of 11514 $(= 3 \times 19 \times 202)$.

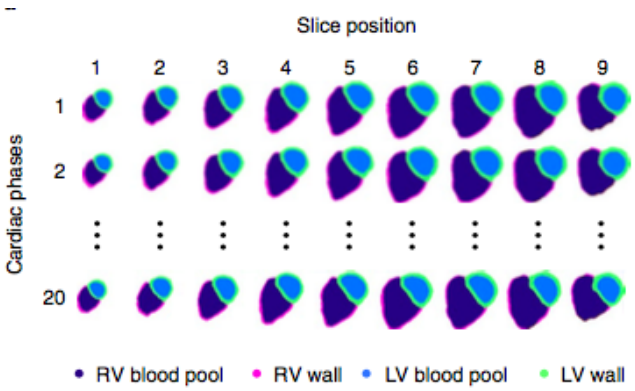


Figure 3.1.: An example of an automatic cardiac image segmentation of each short-axis cine image from the apex (slice 1) to the base (slice 9) across 20 temporal phases (figure 1(a) in Bello et al. (2019)).

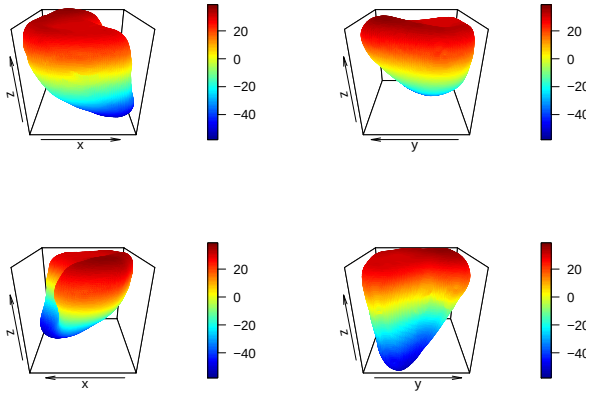


Figure 3.2.: A visualization of a patient's right ventricular at a specific temporal phase. See from four different directions.

3.2. Low dimensional clinical factors and the survival data

The clinical factors $\boldsymbol{y}$ consists of 8 covariates:

- RV end-diastolic volume (RV1, the quantity of blood in the right ventricle at the end of diastole, just before systole starts).
- RV end-systolic volume (RV2, the volume of blood in the right ventricle at the end of contraction).
- the difference between these measures expressed as a percentage of RV1, RV ejection fraction (RV3).
- age, which has been transformed to a continuous variable.
- sex, 1 for male and 0 for female.
- six-minute walk distance (SMWD), with larger values indicating healthier individuals.
- functional class (FC), with value 1, 2, 3 and 4. Larger values mean severer symptom of pulmonary hypertension.
- mean pulmonary artery pressure (MPAP), a direct measure for pulmonary hypertension.

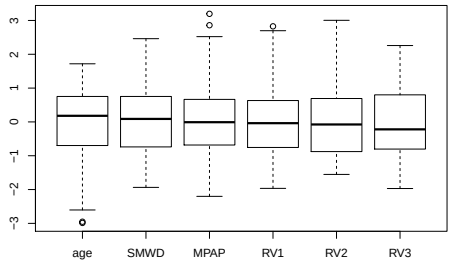
The first three are computed from MRI images, and their normal values should be within certain intervals. The latter five are clinical risk factors which are recorded by doctors during diagnosis, with the aid of other medical examinations. See table 3.1, 3.2 and figure 4.2 for a summary. It is found the data is in pretty good quality, as few extreme values involved and balanced proportion in sex. The exception is functional class, for the most patients being in class 3.

	age	SMWD	MPAP	RV1	RV2	RV3
Min	19.53	20.0	18.0	72.0	33.0	11.0
Max	87.80	576.0	82.0	369.0	303.0	69.0
Mean	62.87	264.8	44.1	193.8	125.0	38.0
Median	65.49	276.0	44.0	191.5	120.5	35.0
Variance	210.5316	15988.0422	140.6284	3842.8875	3511.3488	188.2334

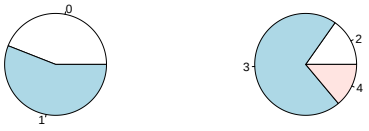
Table 3.1.: Data description for quantitative variables.

Sex		FC	
0(Female)	133	Class 1	
1(Male)	169	Class 2	46
		Class 3	214
		Class 4	42

Table 3.2.: Data description for qualitative variables.



(a) The data has been standardized before plotting.



(b) Left: for sex (1 = "male"; 0 = "female"); right: for functional class.

Figure 3.3.: Data visualization for clinical factors.

The survival data is in the form of $(t_i, \delta_i)_{i=1}^N$ - t_i for ith subject's survival/censoring time and δ_i for his/her censoring status ("$\delta_i = 0$" indicates censoring). The basic Kaplan-Meier estimate for survival function is plotted in figure 3.4. The estimated survival function evenly decreases almost for the entire time line.

To preliminarily evaluate the effect of qualitative variables, sex and functional class, Kaplan-Meier plots and log rank test are adopted. Figure 3.5 shows Kaplan-Meier plots for subjects with different genders and different functional classes, implying that sex does not have much influence on the survival outcome, while functional class does. The result of log rank test (see Appendix A.2) for the equivalence of survival functions in different groups, also proves the point - $p = 0.53$ for that in sex and $p < 0.001$ for that in functional class.

For the effect of quantitative variables, univariate Cox proportional hazard regression is conducted for each factor. Table 3.3 lists the results and shows that apart from mean pulmonary artery pressure and RV end-diastolic volume, other factors significantly (p-value) influence the patients' risk. However numerically, the impacts of RV end-systolic volume and six-minute walk distance are tiny. Also, RV ejection fraction is the only one that negatively affects the survival outcome. In fact, some of the results are in conflict with the clinical commons, which exactly indicates the basic Cox model is improper for the data.

	age	SMWD	MPAP	RV1	RV2	RV3
exponential of the coefficient	1.03	1.00	1.01	1.00	1.00	0.98
p value	< 0.005	0.01	0.14	0.06	0.02	<0.005

Table 3.3.: Results of the univariate Cox proportional hazard regressions for quantitative variables.

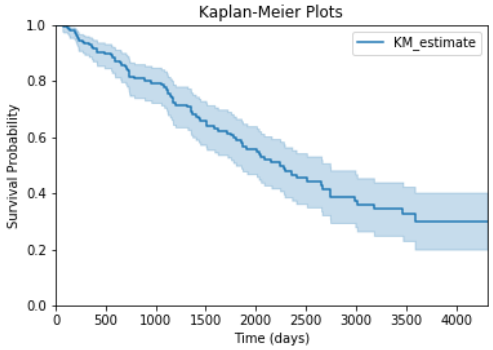
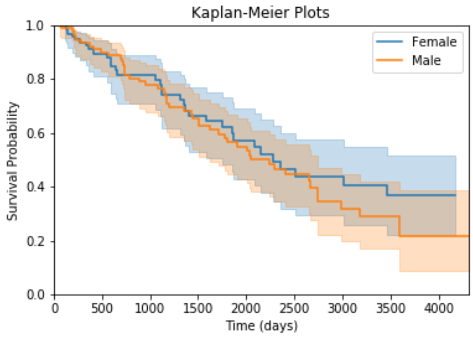
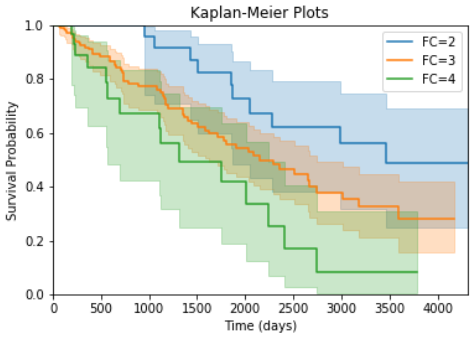


Figure 3.4.: Kaplan-Meier plots for all subjects. The estimated survival function along with its 95 % confidence interval is showed.



(a) Kaplan-Meier plots for subjects with different genders. P-value of log rank test: 0.53.



(b) Kaplan-Meier plots for subjects with different functional classes. P-value of log rank test: < 0.001.

Figure 3.5.: The estimated survival functions along with their 95 % confidence interval are showed.

3.3. Data pre-processing

Data normalization is proved to be able to accelerate the training process and reduce the chances of getting stuck in local optima (Ioffe and Szegedy, 2015). Hence, before being fed to the neural network, all the data are normalized (except for the survival data). For a variable V, Z-score normalization is applied:

$$V^{\star} = \frac{V - mean(V)}{\sqrt{var(V)}}. \tag{3.2}$$

Sex and functional class are also normalized in this way. It is reasonable because it does not change much for sex (after normalization it still takes two possible values), and functional class, specifically, is a ordinal variable, originally having numerical meanings.

4. When low dimensional clinical factors come

In Bello et al. (2019)'s paper, the optimal model for survival prediction is the model only with cardiac motion features. However, empirically speaking, clinical factors can also influence patients' survival curve, e.g. elder patients will have higher risk since their physical mechanism has already degenerated and makes them easier to die. The preliminary analysis in section 3.2 also supports the point.

In this chapter, three approaches to effectively incorporate clinical factors are explored, valid and discussed. The best one accommodates the issue brought by the asymmetry of the two stream of the data and is found to have significant better performance over the optimal model proposed by Bello et al. (2019). The three corresponding extended models are presented in section 4.1, followed by its implementation (section 4.2) and results (section 4.3). Section 4.4 provides discussion on how the clinical factors help the survival prediction.

4.1. Three extended models

In this section, the underlying model (Bello et al., 2019) only inputting cardiac motion data is firstly described in section 4.1.1. The underlying model is a hybrid neural network, consisting of a denoising autoencoder that learns the compressed representation of cardiac motion and a linear-connected layer which handles the prediction task. Based on that, three extensions are proposed in section 4.1.2.

4.1.1. The underlying model

Denoising autoencoder (Vincent et al., 2008) is a widely used feature extraction and dimension reduction technique, which is also referred as the nonlinear version of Principle Component Analysis (PCA). The aim of denoising autoencoder is to unsupervisedly learn compressed representations robust to partial corruption of the input pattern. The input, denoted as $\boldsymbol{x} \in \mathbb{R}^p$, will be corrupted by random dropout and then fed to a hidden layer, the output of which is in turn fed into a central layer. The above procedure can be regarded as an encoding map, $\phi : \mathbb{R}^p \mapsto \mathbb{R}^q$, $q << p$ and this central layer, $\phi(\boldsymbol{x}) \in \mathbb{R}^q$ represents a robust compressed latent code of $\boldsymbol{x}$. The decoding part is just the reverse of the encoder (no corruption, obviously) and we can use another map to denote it, $\psi : \mathbb{R}^q \mapsto \mathbb{R}^p$. Note that the number of units in the two hidden layers in encoder and decoder (see figure 4.1) are the same for the architecture symmetry, and not predetermined, but regarded as optimizable hyper-parameters instead. Also, the size of latent code is a network parameter and should be optimized. The dissimilarity between the input $\boldsymbol{x}$ and reconstructed version is evaluated by l_2 norm (squared error),

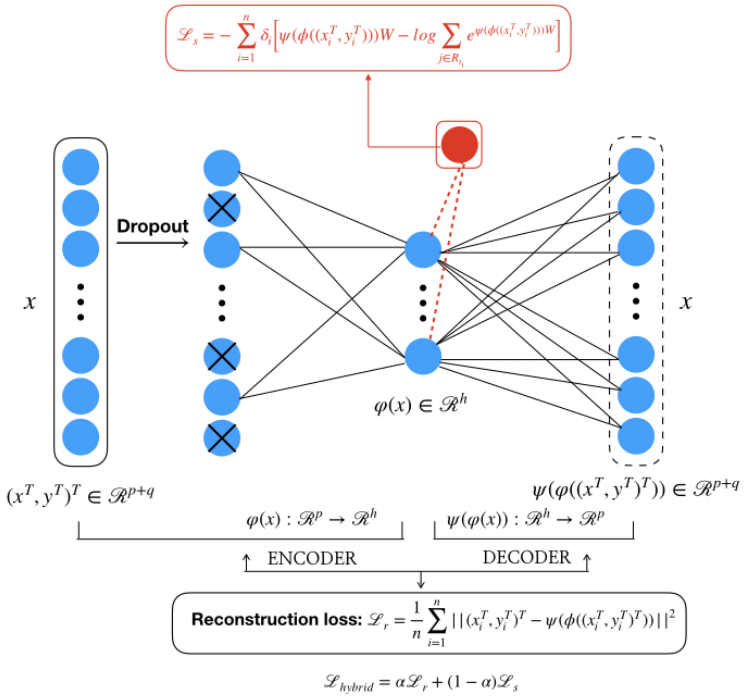


Figure 4.1.: Model 0 - the underlying model. $\varphi(x)$ refers to the latent code. For the sake of simplicity, two hidden layers, one between dropout layer and latent code and the other immediately following the latent code layer, have been excluded from the diagram.

hence leading to the reconstruction loss:

$$L_r = \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{x}_i - \psi(\phi(\boldsymbol{x}_i))||^2, \tag{4.1}$$

where n is the sample size and $||\cdot||$ represents l_2 norm. Minimizing this loss facilitates the denoising autoencoder to learn a latent representation robust to noise.

Apart from reconstruction ability, it is also expected that the learned latent code is capable of survival prediction. So the latent code is then linked to a classical Cox proportional model mentioned in section 2.1, as explanatory covariates. The nature idea is to maximize the likelihood of arising this set of failure time data $\{t_i, \delta_i, \boldsymbol{Z}_i\}_{i=1}^n$ and force the networks to supervisedly learn the relationship between cardiac phenotypes and survival outcomes. So the loss for survival part will be a negative log-partial likelihood:

$$L_s = -\sum_{i=1}^{n} \delta_i \Big[W^T \psi(\phi(\boldsymbol{x}_i) - log \sum_{j \in R_{t_i}} e^{W^T \psi(\phi(\boldsymbol{x}_i))} \Big], \tag{4.2}$$

where W denotes the weights between latent code and prediction layer. Hence, $W^T\psi(\phi(\boldsymbol{x}_i))$ is called i patient's (relative) risk. Then, the overall network is trained via minimizing the hybrid loss:

$$L = \alpha L_r + (1-\alpha)L_s, \quad \alpha \in (0,1), \tag{4.3}$$

where α controls the contribution trade-off between reconstruction and survival prediction, and is also treated as a hyper-parameter. The described network is depicted in figure 4.1.

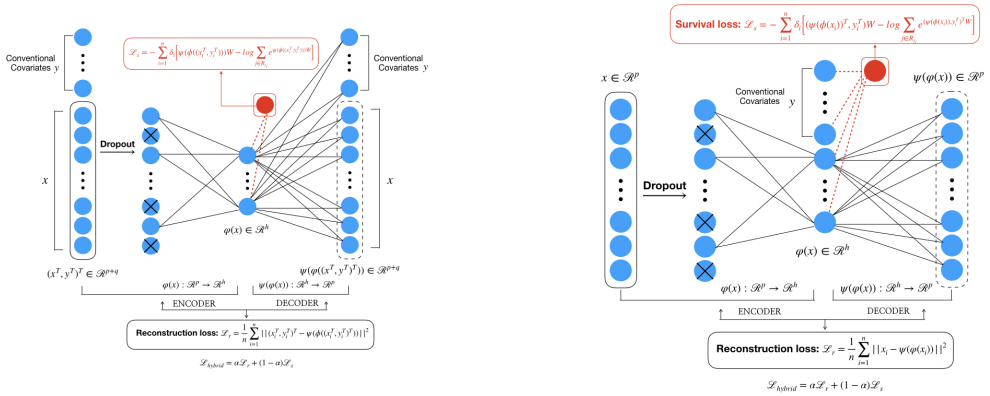
Remark 2. The application of denoising autoencoder here is aimed to reserve the anatomical features in survival prediction and make the latent code more interpretable (Biffi et al., 2018).

4.1.2. The extended models

Based on the above underlying model (figure 4.1), three extensions to incorporate low dimensional clinical factors are proposed.

Model 1 (figure 4.2a) simply adds extra information with input layer and there is corruption as well as reconstruction involved. Model 2 (figure 4.2b) tries to magnify the effect of extra information and adds it with latent code, without corruption and reconstruction. Model 3 (figure 4.3) is inspired from the idea to allow non-linear effect of clinical factors and interaction between them and latent code. So extra information is added with the latent code and one more hidden layer is placed between latent code and prediction layer. The survival part of model 3 is like Faraggi and Simon (1995)'s three-layer artificial neural network now, rather than simply a Cox proportional hazard model.

Remark 3. For the sake of simplicity, two hidden layers in autoencoder have been excluded from the diagram, in figure 4.2 and 4.3.



(a) Model 1: add extra information with input layer and be with corruption and reconstruction.

(b) Model 2: add extra information with latent code and be without corruption and reconstruction.

Figure 4.2.: Less efficient models. For the sake of simplicity, two hidden layers in autoencoder have been excluded from the diagram.

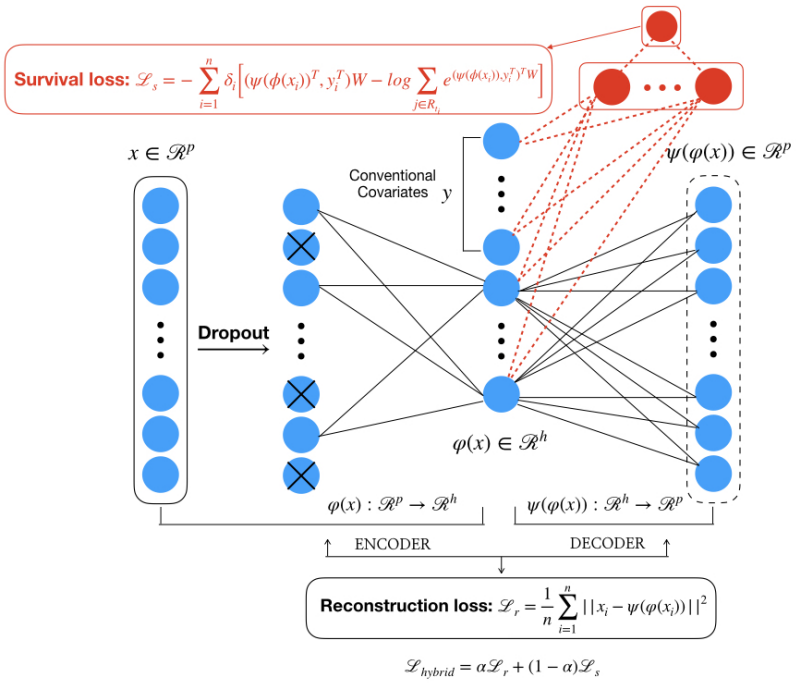


Figure 4.3.: Model 3 - add extra information with latent code and put one more hidden layer between latent code and prediction layer. For the sake of simplicity, two hidden layers in autoencoder have been excluded from the diagram.

4.2. Experiments

Model 0, 1, 2 and 3 were implemented and trained with Python deep-learning libraries Keras (Chollet et al., 2015) and TensorFlow (Abadi et al., 2015), on a high-performance computing cluster with an Intel Xeon E5-1660 CPU and NVIDIA 2080 Ti GPU. The experiment details are listed in this section. The input data are the flattened cardiac motion data described in section 3.1, combined with 8 clinical factors mentioned in section 3.2.

4.2.1. Predictive accuracy metric

Prediction performance is evaluated using Harrell's concordance index (C-index, Harrell et al. (1982)), estimated by

$$\hat{\mathcal{C}} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} I(\hat{h}_i > \hat{h}_j) I(t_i < t_j) I(\delta_i = 1)}{\sum_{i=1}^{m}\sum_{j=1}^{m} I(t_i < t_j) I(\delta_i = 1)}, \quad i, j = 1, \ldots, m, \tag{4.4}$$

where m denotes the number of test data and $\hat{h}_i$ is ith subject's predicted risk. This index is also a survival version of area under the receiver operating characteristic curve and the concordance is discriminated by the criterion that subjects who get sooner death should be predicted higher risk. Note that the multiplication of the indicator $I(\delta_i = 1)$ restricts the summation over informative pairs, in the sense that it guarantees that ith subject dies earlier. So, C-index can be comprehended as the proportion of concordant pairs among all informative pairs.

4.2.2. Hyperparameter tuning

Autoencoder is usually criticized as overfitting. To avoid it, apart from dropout, we employed l_1 regularization (see section 2.2.1 for details) to the weights directing the hidden layer in encoder. Also, ReLu activation function (see section 2.2.1 for details) is applied(except two output layers which use a linear activation), partially to encourage sparsity in the network, especially in latent code. With the adaptive moment estimation (Adam, Kingma and Ba (2014)) algorithm, the network was trained for 100 epochs with a batch size of 16 subjects and the loss function was minimized by back-propagation method (Hecht-Nielsen, 1992).

In our models, hyper-parameters include learning rate (in Adam algorithm), $l1$ regularization parameter, reconstruction loss proportion α, dropout ratio, number of nodes in two autoencoder hidden layers and the size of latent code. For model 3, number of units in survival part's hidden layer is also regarded to be optimizable. In model construction, we applied six-fold cross validation to search the optimal hyper-parameters based on discriminative performance. The searching procedure is conducted by particle swarm optimization algorithm (PSO, Kennedy (2010)), guided with some prior and empirical knowledge. In our experiment, as model 0, 1 and 2 have similar structures, they share the same set of hyper-parameters and model 3 has a separate set of hyper-parameters. The predefined searching ranges and optimized hyper-parameters are listed in table 4.1.

Hyper-parameter	Searching range	for model 0, 1 and 2	for model 3
Dropout	$[0.1, 0.9]$	0.20	0.54
Units1	$[75, 250]$	125	161
Units2	$[5, 20]$	13	16
α	$[0.3, 0.7]$	0.55	0.37
Learning rate	$[10^{-6}, 10^{-4.5}]$	$10^{-5.19}$	$10^{-4.95}$
l_1 penalty	$[10^{-7}, 10^{-4}]$	$10^{-5.91}$	$10^{-5.71}$
Units3	$[5, 20]$		11

Table 4.1.: The predefined searching ranges and optimized hyper-parameters for model 0, 1, 2 and 3. "Units1", "Units2", "Units3" represents the number of units in autoencoder hidden layers, latent code dimensionality and the number of units in survival hidden layers, respectively.

4.2.3. Internal validation

Due to the sample size, it will lose certain efficiency if we divide 302 patients into training set and testing set. So the thesis employs a bootstrap-based optimism-adjusted estimate for the C-index, which is verified to be unbiased on some clinical data sets by Smith et al. (2014). It helps to simulate the external cases by bootstrapping and provides the model's generalization accuracy only using the training samples. Details can be found in chapter 17, Efron and Tibshirani (1994) and here outlines the workflow.

Suppose there are data $\mathcal{X} = \{\boldsymbol{X}_i\}_{i=1}^N$ and N is the data size. The prediction accuracy metric C-index can be regarded as a function of the data $\mathcal{X}$ and a model, $\eta(\cdot)$, i.e. $\hat{\mathcal{C}}(\mathcal{X}, \eta(\mathcal{X}))$ means the C-index computed from $\mathcal{X}$ using the model trained by $\mathcal{X}$.

The bootstrap-based optimism-adjusted estimates for C-index can be derive as followed,

1. Develop a model based on the full training sample, $\mathcal{X}$, and compute the C-index on the same full data, yielding the apparent accuracy:

$$\hat{\mathcal{C}}_{app} = \hat{\mathcal{C}}(\mathcal{X}, \eta(\mathcal{X})).$$

2. Develop a model based on a bootstrapped sample, $\mathcal{X}^{\star b}$ (resampled from $\mathcal{X}$ with replacement), and compute the C-index on both the same bootstrapped sample and the full data, $\mathcal{C}(\mathcal{X}^{\star b}, \eta(\mathcal{X}^{\star b}))$ and $\mathcal{C}(\mathcal{X}, \eta(\mathcal{X}^{\star b}))$, respectively. Then, the optimism is calculated by their difference:

$$\hat{\mathcal{C}}^b_{opt} = \hat{\mathcal{C}}(\mathcal{X}^{\star b}, \eta(\mathcal{X}^{\star b})) - \hat{\mathcal{C}}(\mathcal{X}, \eta(\mathcal{X}^{\star b})).$$

3. Repeat step 2 over $b = 1, \ldots, B$, B is a preset bootstrapping time.

4. Average across the optimism estimates from step 2 to 3 and subtract the resulting quantity from the apparent C-index derived from step 1, leading to the optimism-adjusted estimate:

$$\hat{\mathcal{C}}_{adj} = \hat{\mathcal{C}}_{app} - \frac{1}{B}\sum_{b=1}^{B} \hat{\mathcal{C}}^b_{opt}. \tag{4.5}$$

Remark 4. In practice, the optimal hyper-parameters found based on the full data can be used in all model trainning, whenever in step 1 or 3. There is no necessary to optimize hyper-parameters for every time according to the learning logic that the network parameters should not change with same type of data.

Remark 5. In the computation of bootstrap-based optimism-adjusted C-index, the optimism is stable since it is an average over B numbers, while the apparent one may vary time by time due to the randomness in model training. To solve the issue, it is recommanded to calculate multiple apparent C-indexes and take the average.

Furthermore, after bootstrapping, the confidence intervals can be produced simultaneously, based on Efron's percentile method (Efron and Hastie, 2016). This method directly uses the quantile of their distribution to define percentile confidence limits. In the experiment, a confidence interval $[c^{\star 1}, c^{\star 2}]$ is found such that

$$\hat{P}(c^{\star 1} < \mathcal{C}^{\star} < c^{\star 2}) = 1 - \alpha,$$

given the significance level of α. In the above equation, $\hat{P}$ refers to the probability measure for sampling distribution and $\mathcal{C}^{\star}$ is derived from the bootstrap samples. There is some flexibility in choosing $c^{\star 1}$ and $c^{\star 2}$. By default, $c^{\star 1}$ will be chosen to satisfy

$$\hat{P}(c^{\star 1} > \mathcal{C}^{\star}) = \frac{\alpha}{2}.$$

For symmetrical sampling distributions, this method works comparably well, even better than studentized bootstrap interval (Efron and Hastie, 2016), as the shape of the bootstrap distribution is considered in Efron's percentile method. On the contrary, this method performs poorly for skewed distributions.

Remark 6. The reason for not using cross-validation to estimate the prediction error partly lies in avoiding underestimate due to hyper-parameter optimization, which also targets at discriminative performance. The other reason, obviously, is taking the randomness of the data into consideration and derive a more realistic estimate for the population.

Remark 7. Bootstrapping was replicated for 500 times in this experiment.

4.2.4. Model comparison

Model comparison is conducted by pair-wise two sample test in bootstrap paradigm. Suppose there are two models M_1 and M_2, and data $\mathcal{X} = \{\boldsymbol{X}_i\}_{i=1}^{N}$ with N the size of the data set. The prediction accuracy metric C-index can be regarded as a function of the data $\mathcal{X}$, $\mathcal{C}_1(\mathcal{X})$ and $\mathcal{C}_2(\mathcal{X})$, respectively for model M_1 and M_2. To test whether model M_2 outperforms model M_1, the hypothesis test can be set as followed:

$$H_0 : \mathcal{C}_1(\mathcal{X}) \geq \mathcal{C}_2(\mathcal{X}) \longleftrightarrow H_1 : \mathcal{C}_1(\mathcal{X}) < \mathcal{C}_2(\mathcal{X}),$$

which is equivalent to $H_0 : \mathcal{C}_1(\mathcal{X}) = \mathcal{C}_2(\mathcal{X}) \longleftrightarrow H_1 : \mathcal{C}_1(\mathcal{X}) < \mathcal{C}_2(\mathcal{X})$. Using the data $\mathcal{X}$, B sets of bootstrap samples can be generated from it, denoted as $\mathcal{X}^{\star 1}, \ldots, \mathcal{X}^{\star B}$. Then

$\mathcal{C}$ can be estimated by equation 4.4, leading to the approximation for the p-value under H_0:

$$\begin{aligned} \hat{p} &= \frac{1+\sum_{i=1}^{B} I(\hat{\mathcal{C}}_1(\mathcal{X}^{\star i}) - \hat{\mathcal{C}}_2(\mathcal{X}^{\star i}) \geq \hat{\mathcal{C}}_1(\mathcal{X}) - \hat{\mathcal{C}}_2(\mathcal{X}))}{B+1} \\ &= \frac{1+\sum_{i=1}^{B} I(\hat{\mathcal{C}}_1(\mathcal{X}^{\star i}) \geq \hat{\mathcal{C}}_2(\mathcal{X}^{\star i}))}{B+1} \end{aligned} \tag{4.6}$$

This approximated p-value ensures that the test is a conservative test and statistical judgement can be made by certain confidence level.

In the experiment, model comparison can be completed at the same time as internal validation. For external generalization, it is better to use optimism-adjusted estimate for $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$. Then in formula 4.6, $I(\hat{\mathcal{C}}_1(\mathcal{X}^{\star i}) \geq \hat{\mathcal{C}}_2(\mathcal{X}^{\star i})) = I(\hat{\mathcal{C}}_1^{app} - \hat{\mathcal{C}}_1^{opt}(\mathcal{X}^{\star i}) \geq \hat{\mathcal{C}}_2^{app} - \hat{\mathcal{C}}_2^{opt}(\mathcal{X}^{\star i}))$, following the notation in section 4.2.3.

4.3. Results

model	C-index	95% bootstrap confidence interval
Model 0	0.7979	(0.7536, 0.8373)
Model 1	0.7978	(0.7522, 0.8402)
Model 2	0.7998	(0.7564, 0.8371)
Model 3	0.8276	(0.7837, 0.8681)
Model 1 with noise	0.8068	(0.7644, 0.8457)
Model 2 with noise	0.8012	(0.7548, 0.8407)
Model 3 with noise	0.8087	(0.7718, 0.8433)

Table 4.2.: Model 0 represents the model without new clinical factors. Models with noise stand for replacing clinical factors with standard gaussian noise and use the same hyper-parameters with the corresponding models. The C-index is computed after optimism adjustment and 95% confidence intervals are built by bootstrap.

	model 0 v.s. model 1	model 0 v.s. model 2	model 0 v.s. model 3
p-value	0.5049	0.4554	0.0396

Table 4.3.: Model comparisons based on 500 sets of bootstrapping samples.

As we can see in table 4.2 and 4.3, model 1 and 2 have no significant over-performance compared with model 0, while model 3 has better prediction ability than model 0. The latter is also verified by bootstrapping pair-wise two sample test described in section 4.2.4 , with p-value 0.0396 and being significant at 0.05 confidence level. Also, in model 1 and 2, the clinical factors are found to behave like standard gaussian noise.

To explain the model comparison visually, the differences between estimated optimism-adjusted C-indexes among the four models are computed and plotted in figure 4.4 for each set of bootstrap samples. For clear visualization, only 100 sets of bootstrap samples

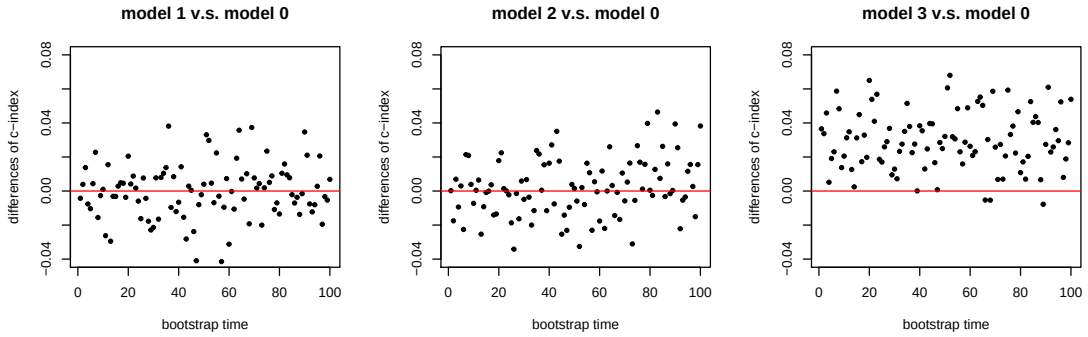


Figure 4.4.: The differences between estimated C-indexes among the four models. Each black point stands for the difference in a set of bootstrap samples, and the red line marks $y = 0$. 100 of 500 sets of bootstrap samples are randomly selected for the plot.

are selected to make up for the plot. In figure 4.4, black dots are almost evenly distributed at the two sides of the red line $y = 0$, indicating similar performance between model 0 and model 1, model 2. While in the third the sub-figure, majority dots live in the upper of the red line, demonstrating the dominance of model 3 over model 0.

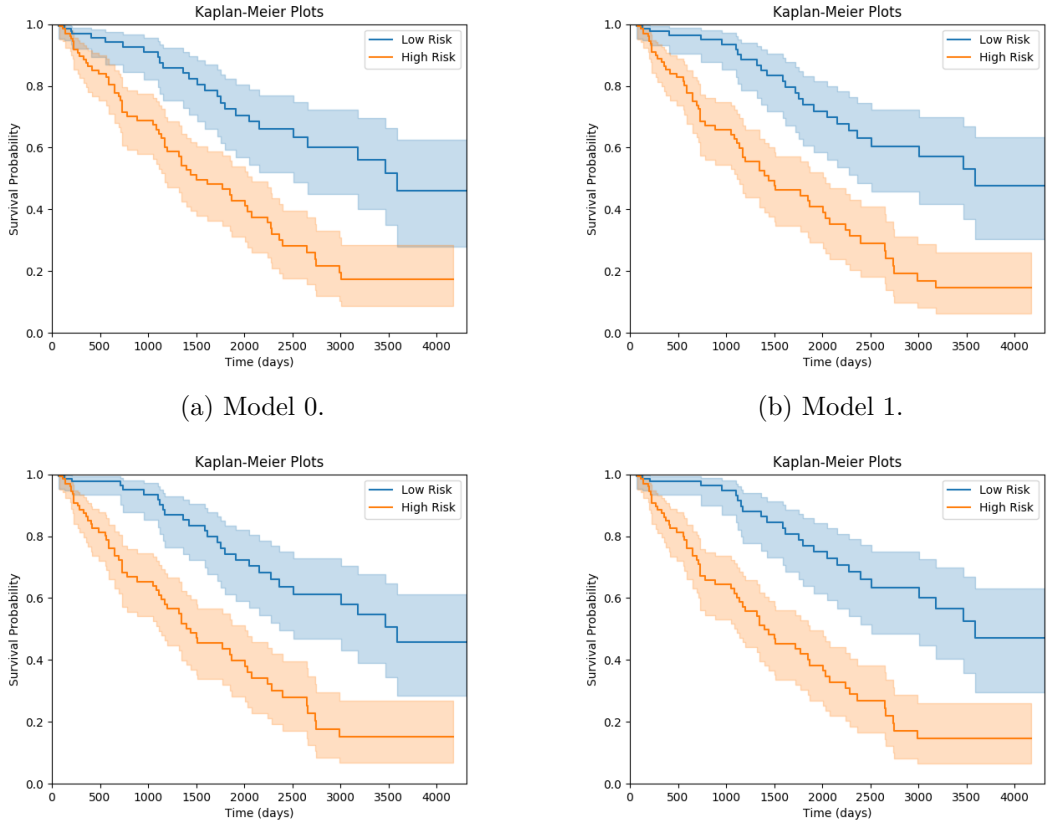
Moreover, the experiments that replace clinical factors with standard gaussian noise show that the outperformance of model 3 is not due to overfitting or other issues from the networking training process, but it is the extra clinical factors that provide information and make effect in survival prediction. Though the improvement is numerically not large, statistically significant indeed. Kaplan-Meier plots in figure 4.5 also implies this point, as model 3 divides two risk groups more separately (The group division is based on individuals' predicted risk and discriminated by the median value of the risk).

Remark 8. Incorrect model comparison may be concluded from the argument that their 95% bootstrap confidence intervals are all overlapped, so model 1, 2 and 3 do not significantly outperform model 0. Note that the confidence intervals listed in 4.2 are constructed from different bootstrap samples, while it should be a pair-wise two sample test problem - comparing the models on the same set of bootstrap samples.

4.4. Discussion

The research shows that model 1 is not a reasonable approach to add more information in denoising autoencoder-based network. In the experiment, although theoretically, the reconstruction of cardiac motion data and clinical factors together allows the autoencoder to learn hybrid representation of these two streams of information, it turns out that the new data is just regarded as a special corruption and the denoising autoencoder is robust to it: augmenting small volume of data will not destroy the stability of latent code.

Model 2 is inspirational. Clinically speaking, those factors should influence patients' sur-



(a) Model 0.

(b) Model 1.

(c) Model 2.

(d) Model 3.

Figure 4.5.: Kaplan-Meier plots for subjects with different risk levels in four models. The group division is based on individuals' predicted risk and discriminated by the median value of the risk. The estimated survival functions along with their 95 % confidence interval is showed.

vival, as mentioned at the beginning of the chapter. However, directly combined with the latent code, they seem to loss the effect for survival prediction. In the following section, analysis deepens into understanding the explanatory behaviours of clinical factors in model 2. And the last section will discuss adaptions of model 3 for radiogenomics research.

4.4.1. Model interpretation

In this section two questions are addressed: **in linear manners** (they linearly contribute to the prediction payer in model 2),

- whether the clinical factors has learnt information from extra data?
- whether the clinical factors provides information for survival prediction?

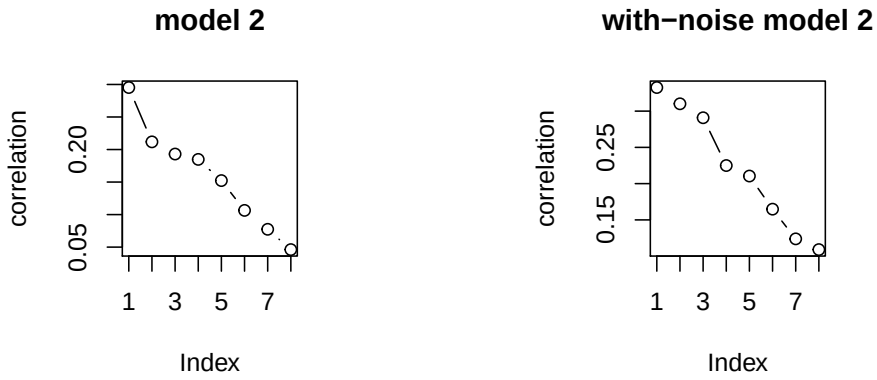
The both answers are negative.

For the first one, the latent code in model 2 is extracted and the preliminary correlation analysis shows that the Pearson correlation coefficients between all pairs of variables from latent code and clinical factors are below 0.2, which is the same level of correlations between noise and latent code in with-noise model 2.

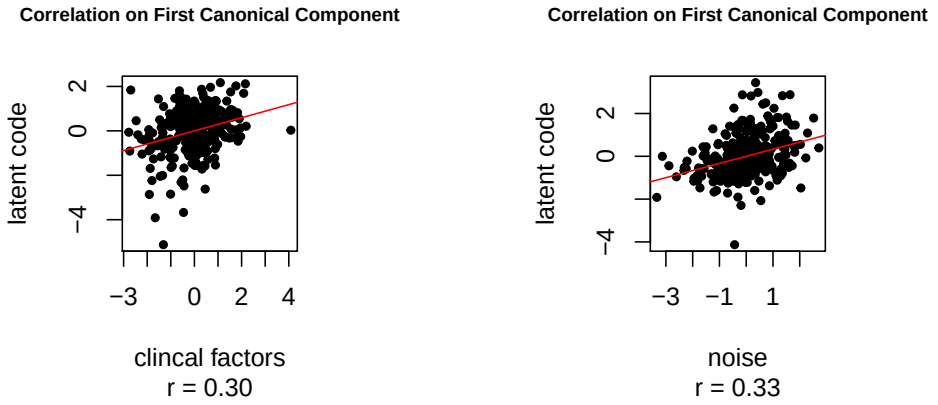
To evaluate the overall linear correlation (or potential correlation after linear combination) between the two sets of variables, latent code and clinical factors, Canonical Correlation Analysis (CCA, Hotelling (1992)), a powerful multivariate analysis tool, is properly used. The basic idea is to find linear combinations of the latent code and clinical factors, respectively, so that their correlation is maximized afterwards. This pair is called first pair of canonical variables. The second pair of canonical variables are sought via the same maximization subject to the constraint that they are to be uncorrelated with the first pair of canonical variables. The latter pairs of canonical variables can be derived in the same way. Similar to principle component analysis (PCA), the first one or two pairs of canonical variables will explain the most linear correlation between the two sets of variables.

After applying CCA to model 2 and with-noise model 2, figure 4.6a indicates that the correlations in each pairs of canonical variables do not vary much. Also, the linear correlations between first two canonical variables are visualized in figure 4.6b and 4.6c. It is found that the overall linear correlation between latent code and clinical factors is no more than that of noise and latent code. So the above correlation analysis concludes that cardiac motion data learns nothing about clinical factors in linear manners.

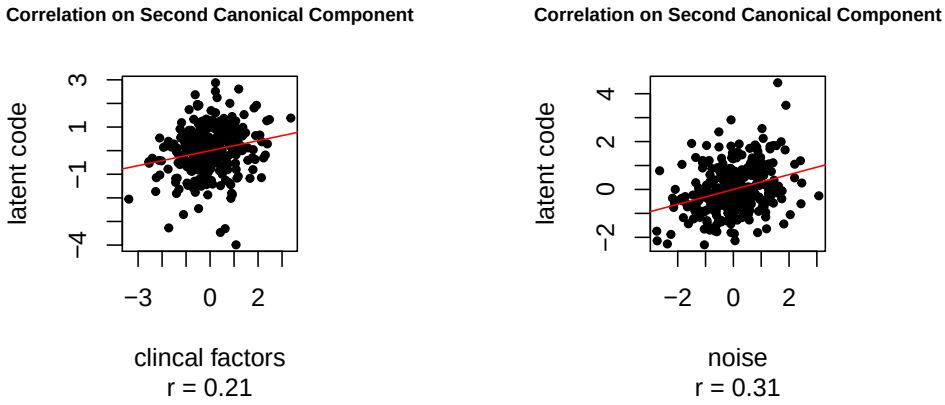
For the second question, whether clinical factors can linearly contribute to survival prediction is evaluated. Linear regressions are further conducted between the predicted risk and clinical factors, as well as between the predicted risk and latent code, in both two models. The Table 4.4 shows the R-squared statistic, a measure for model fitting, which implies latent code explains almost all about the predicted risk and clinical factors have similar effect as noise for survival prediction.



(a) Correlations between pairs of canonical components. X-axis represents the order of pairs.



(b) For model 2



(c) For model 2 with noise.

Figure 4.6.: For (b) and (c): black points - individuals represented by first two pairs of canonical components. Red line - fitted straight line by linear regression. r denotes the Pearson correlation between x-axis variable and y-axis variable.

model	predicted risk v.s. clinical factors	predicted risk v.s. latent code
Model 2	0.02	0.84
Model 2 with noise	0.03	0.95

Table 4.4.: R-squared statistics for regression models.

All in all, clinical factors behave like gaussian noise in the network processing from a linear perspective. This inspires us to explore the nonlinear effect of the clinical factors and its interaction with latent code. This is why model 3 is come up. Nevertheless, it is still hard to quantify this nonlinear effect and interaction, while the outperformance of model 3 proves the rationale.

4.4.2. Model adaption

Radiogenomics is a new medical research direction in recent years, which focuses on the relationship between imaging phenotypes and genomics (Mazurowski, 2015). It is desirable to introduce knowledge of genetic variants to survival networks to improve outcome prediction by jointly analysing cardiac motion traits with inheritable risk factors. For the new challenge, model 3 can be evolved further to adapt to handling high dimensional new information, like genetic data. Unlike directly combining low dimensional clinical factors with latent code, pre-processing of the new genomics data is recommanded, i.e. conducting another denoising autoencoder to find compressed representation of genotypes. The key thing is that hidden layers are required after merging the two latent codes, to allow the nonlinear effects and interaction of them.

5. A new type of prediction

In Bello et al. (2019)'s paper, they input the compressed latent code directly to the canonical Cox proportional hazard model. As pointed by De Laurentiis and Ravdin (1994), Burke and MacKenzie (2017) and so on, proportional hazard may not always hold. Lee et al. (2018) suggested to relax this assumption and to estimate the probability density instead, in which there are no concrete underlying models. Actually, this proposal is more like a return, as the ancient neural networks with survival analysis originally care about the density function or the survival function. But at that time, they just regarded it as a classification problem and unconsciously got rid of Cox model.

In this chapter, the survival prediction part of model 3 (in chapter 4) is altered in section 5.2 and an empirical likelihood type loss function is introduced in section 5.1. In this case, the predicted risk is not a constant any more and varies in time. So a new time-varying evaluation metric is employed (section 5.3). Section 5.4 shows the experiment details and the results, and section 5.5 is for discussion.

5.1. Empirical likelihood type loss function

It has been pointed out that letting the neural networks to learn the survival function or the cumulative distribution function may lead to a non-monotone function. So probability density is chosen to be the output of the neural network and the normalising requirement potentially helps the learning process. Strictly, the idea is borrowed from Lee et al. (2018) who considered that the presuming concrete forms of the link between hazard function and covariates was too constricted and tried to harness the flexibility of network to learn their relationship.

In particular, the survival time X can be discretized into K time slots $\{l_1, l_2, \ldots, l_K\} = \mathcal{T}$. Assume that length of all the slots are finite (human's life is finite anyway). Denote $p_1 = P(X \in l_1), p_2 = P(X \in l_2), \ldots, p_K = P(X \in l_K)$. Now the output of the network is the discretized distribution of the survival time $(p_1, p_2, \ldots, p_K)$. Hence, given the survival data $\{t_i, \delta_i\}_{i=1}^N$, the negative log-empirical likelihood is introduced as the loss function,

$$L = -\sum_{i=1}^{N}[I(\delta_i = 1) \cdot log(p_{k:t_i \in l_k}) + I(\delta_i = 0) log(1 - \sum_{j=1}^{k:t_i \in l_k} p_j)]. \tag{5.1}$$

The underlying reason is maximizing likelihood - construct a model to be most likely to acquire this set of data. For a censored subject, the probability of being observed the censoring time is the total probability assigned to the following intervals; for a non-censored subject, the probability of being observed the survival time is just the probability of the corresponding interval that the time falls into. Here is an example to explain the novel

loss.

Example 9. Suppose that survival time T can take values from $[0, 10]$, then we divide $[0, 10]$ into 10 sub-interval: $[0, 1), [1, 2), \ldots, [8, 9), [9, 10]$, and the corresponding probabilities are $p_1, p_2, \ldots, p_{10}$. There are three subjects: one dies at time 7.5, and the other two survives to 6.6 and 8.7, respectively, and gets censored. Then the loss for them is:

$$\begin{aligned} L = & - [1 \times log(p_8) + 0 \\ & + 0 + 1 \times log(1 - \sum_{i=1}^{7} p_i) \\ & + 0 + 1 \times log(1 - \sum_{i=1}^{9} p_i)] \\ = & - [log(p_8) + log(p_8 + p_9 + p_{10}) + log(p_{10})]. \end{aligned}$$

Remark 10. In practice, l_K should be partitioned such that no subjects get censored in l_K, otherwise the loss for that subject is not well defined (have to take log(0)).

5.2. The model with the new loss

This new loss is applied to model 3 with the same cardiac motion data and eight clinical factors, and hoped to improve the survival prediction. With the new survival estimate goal, the adapted model, model 4, can be seen in figure 5.1. A denoising autoencoder is firstly used to find robust compressed representation for cardiac motion. Then, combined with clinical factors, the latent code is fed to a softmax layer (linked with a softmax activation function detailed in section 2.2.1), intermediated by a hidden layer to allow nonlinear relationship and optimised interaction. The softmax activated layer can ensure regularization property for probability distribution.

Remark 11. The new model 4 does not ensure better performance. Maybe the proportional hazard assumption holds for the data. Whereas, with proper training, it at least can achieve the similar performance to model 3.

Remark 12. For the sake of simplicity, two hidden layers in autoencoder and one hidden layer in survival part, have been excluded from the diagram, in figure 5.1.

5.3. A weighted time-varying evaluation metric

Prediction performance is also evaluated using Harrell's concordance index (C-index, already interpreted in section 4.2.1), in which the basic idea is that the subjects with higher risk should get to the event earlier among comparable pairs.

To accommodate the issue that Harrell's C-index is only computed at the end of the observation period and just suitable for time-invariant case, Antolini and Biganzoli (2005)

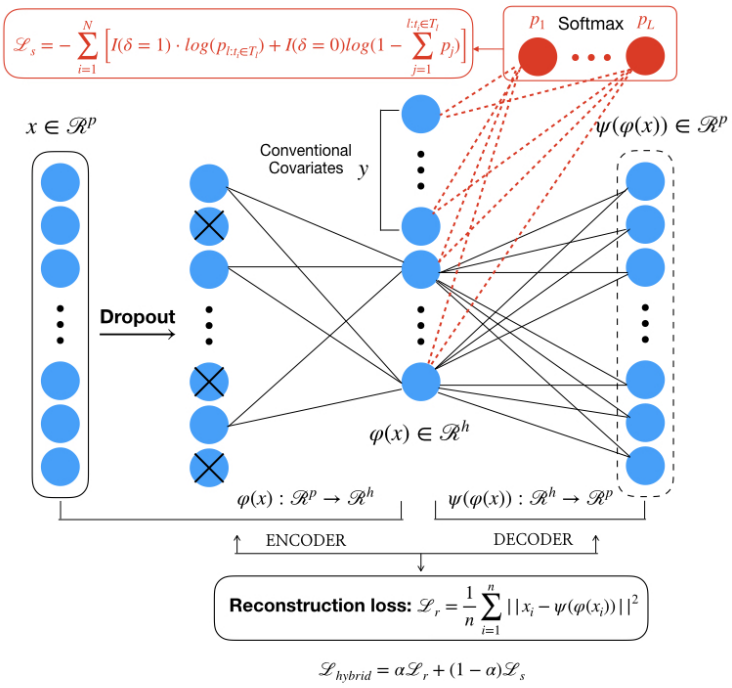


Figure 5.1.: Architecture of the model 4 with a new loss. For the sake of simplicity, two hidden layers in autoencoder and one hidden layer in survival part, have been excluded from the diagram.

proposed a time-varying C-index:

$$\hat{\mathcal{C}}(t) = \frac{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} I(\hat{F}_n(t|\boldsymbol{Z}_i) > \hat{F}_n(t|\boldsymbol{Z}_j))I(t_i < t_j)I(t_i \leq t, \delta_i = 1)}{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} I(t_i < t_j)I(t_i \leq t, \delta_i = 1)}, \tag{5.2}$$

where n is the number of training subjects, m is the number of test subjects and $\hat{F}_n(\cdot)$ is the estimated cumulative distribution function $\hat{F}_n(t) = \sum_{i=1}^{k:t\in l_k} \hat{p}_i$, in which $\hat{p}_i$ is the probability mass estimated by the network. $I(t_i \leq t, \delta_i = 1)$ is to ensure that the pair of i and j is comparable.

Specifically, it is an estimate of the true C-index defined as $\mathcal{C}_n(t) = E_{ij}[I(F_n(t|\boldsymbol{Z}_i) > F_n(t|\boldsymbol{Z}_j))|t_i < t_j, t_i \leq t, D_n^{train}]$, $i, j \in \{1, \ldots, n\}$. The expectation is taken over all pair of subjects in the training set. However, Gerds et al. (2013) pointed out that $\hat{\mathcal{C}}(t)$ was not a consistent estimator and proposed an inverse probability of censoring weighted (IPCW) estimators to correct censoring bias:

$$\hat{\mathcal{C}}_{ipcw}(t) = \frac{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} I(\hat{F}_n(t|\boldsymbol{Z}_i) > \hat{F}_n(t|\boldsymbol{Z}_j))I(t_i < t_j)I(t_i \leq t, \delta_i = 1)\hat{W}_{ij}^{-1}}{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} I(t_i < t_j)I(t_i \leq t, \delta_i = 1)\hat{W}_{ij}^{-1}}, \tag{5.3}$$

where $\hat{W}_{ij}^{-1} = \hat{P}(C > t_i|\boldsymbol{Z}_j)\hat{P}(C > t_i - |\boldsymbol{Z}_i)$ and C is the censoring time.

Remark 13. $\hat{S}_C(t) = \hat{P}(C > t|\boldsymbol{Z})$ is the estimated conditional survival function of the censoring time. $\hat{P}(C > t - |\boldsymbol{Z}) = \hat{S}_C(t-)$ is the left limit of the estimated conditional survival function at time t.

Remark 14. If we know the censoring model or have a consistent estimate for the conditional survival function, $\hat{\mathcal{C}}_{ipcw}(t)$ will converge to $\mathcal{C}_n(t)$ in probability as m goes to infinity, at least point-wisely.

Remark 15. Without no prior knowledge or to simplify the model, the conditional survival function is assumed to be independent of the covariates, i.e. $P(C > t|\boldsymbol{Z}) = P(C > t)$. In this way, $\hat{P}(C > t|\boldsymbol{Z})$ can be simply derived by Kaplan-Meier estimate, but turning the censoring indicator δ around, i.e. $\delta = 1$ for censoring and $\delta = 0$ otherwise.

5.4. Experiments and results

Similar to the experiments in section 4, the model 4 is implemented on the same software and equipments, with high dimensional cardiac data and eight clinical factors. Like the training process described in section 4.2.2, regularization techniques, including dropout and l_1 penalty on the weights directing the hidden layer in encoder are employed. ReLu activation functions are applied as well (except the output layer in decoder for linear activation and the output layer in survival part for softmax activation). Again, the network is trained for 100 epochs with a batch size of 16 subjects by Adam algorithm.

Now apart from the model hyper-parameters listed in table 4.1, the number of units in softmax layer and the length of the study time line should be considered as well. Few literatures mention the maximum years that a patient with pulmonary hypertension can survive to after being diagnosed. But it has been deemed an increase since the

development of specialized therapies for pulmonary hypertension (Sitbon et al., 2014), beginning with the introduction of Flolan in 1995. After consulting professional clinicians, the maximum life is set to 6000 days (about 16 years), started from the day of being diagnosed. Also, with a base of 30 days (approximately 1 month), the the whole time line is divided into 200 intervals, meaning that there are 200 units in the softmax layer.

The hyper-parameters are tuned by six-fold cross validation targeting at the inverse probability of censoring weighted C-index, $\hat{\mathcal{C}}_{ipcw}(t)$, described in the last section. $\hat{\mathcal{C}}_{ipcw}(t)$ is varying in time and it is hard to compare two functions. Borrowing the conduct in Lee et al. (2019), the thesis summarizes the function into a number, $\bar{\mathcal{C}}$, which is the average over $\hat{\mathcal{C}}_{ipcw}(t_{0.25})$, $\hat{\mathcal{C}}_{ipcw}(t_{0.5})$ and $\hat{\mathcal{C}}_{ipcw}(t_{0.75})$, where t_α denotes the lower α sample quantile in observed survival/censoring time $\{t_i\}_{i=1}^N$. Hence, the optimal hyper-parameters are searched to maximizing $\bar{\mathcal{C}}$. As the output of survival part is quite large, the searching range of latent code dimensionality is adjusted to $[20, 50]$. The optimization result is shown in table 5.1.

To be comparable, model 3 with the new C-index was also implemented. Specifically, in equation 5.3, the estimated cumulative distribution function $\hat{F}_n(t)$ is replaced by the predicted relative risk in model 3 of chapter 4. Since the discriminative metric has changed, the model parameters are optimized again and listed in table 5.1 as well. In the following, all mentioned model 3s refer to the model 3 with the new time-varying C-index, rather than the model 3 in chapter 4.

Hyper-parameter	Searching range	model 3	Searching range	model 4
Dropout	$[0.1, 0.9]$	0.40	$[0.1, 0.9]$	0.40
Units1	$[75, 250]$	125	$[75, 250]$	160
Units2	$[5, 20]$	15	$[20, 50]$	31
α	$[0.3, 0.7]$	0.47	$[0.3, 0.7]$	0.52
Learning rate	$[10^{-6}, 10^{-4.5}]$	$10^{-5.49}$	$[10^{-6}, 10^{-4.5}]$	$10^{-5.65}$
l_1 penalty	$[10^{-7}, 10^{-4}]$	$10^{-4.83}$	$[10^{-7}, 10^{-4}]$	$10^{-4.86}$
Units3	$[5, 20]$	10	$[75, 125]$	100

Table 5.1.: The predefined searching ranges and optimized hyper-parameters for model 3 and 4, with respect to a new weighted time-varying evaluation metric. "Units1", "Units2", "Units3" represents the number of units in autoencoder hidden layers, latent code dimensionality and the number of units in survival hidden layers, respectively. The searching ranges are slightly different between model 3 and 4, due to their discrepant structures.

Due to the data size, bootstrap-based internal validation is applied, like that in chapter 4. Model 3 and 4 are also compared by a bootstrap pair-wise two sample test. The bootstrap procedure is replicated for 500 times. The result, listed in table 5.2, indicates that the model 4 achieves about 3% better prediction performance than that of model 3, and it is significant at the level of 0.1. The model comparison is also visualized in figure 5.2. To make it clear, only 100 sets of bootstrap samples are selected to plot the black dots, which stand for the differences between the estimated optimism-adjusted C-index.

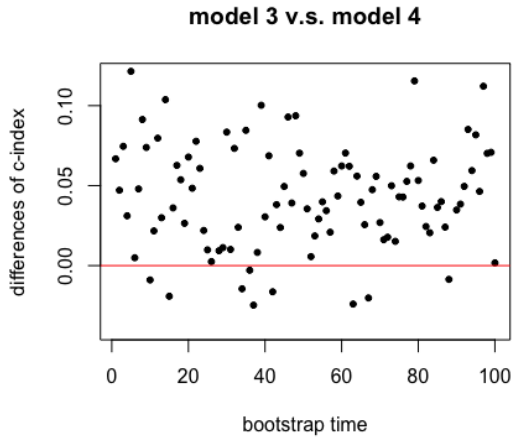


Figure 5.2.: The differences between estimated C-indexes between model 3 and model 4. Each black point stands for the difference in a set of bootstrap samples, and the red line marks $y = 0$. 100 of 500 sets of bootstrap samples are randomly selected for the plot.

It is seen that most of the dots distribute above the red line $y = 0$, demonstrating the better performance of model 4.

Nevertheless, compared with the results of model 0, 1, 2 in table 4.2, table 5.2 shows that model 3 and model 4 require larger 95% confidence intervals, implying that models with new evaluation metric and new survival prediction structure are less stable to some extent. This is because the time-varying C-index is only computed at some time points and it is impossible to make use of the information after the oldest time point.

model	C-index	95% bootstrap confidence interval	p-value
Model 3	0.8272	(0.7645, 0.8852)	0.0990
Model 4	0.8679	(0.7732, 0.9361)	

Table 5.2.: Prediction performance for model 3 and model 4. The C-index is computed after optimism adjustment and 95% confidence intervals are built by bootstrap. Bootstrap pair-wise two sample test implies the significance at 90% confidence level.

5.5. Discussion

As showed in the above experiments and the results, this new type of prediction is appealing in neural networks. Without the restriction of the underlying assumption, the survival time X is simply and flexibly characterized by the deep-learning model. Someone may doubt about the confidence level of the statistical test and argue that model 3 will behave better averagely for one time in ten experiments. Despite of this, the new type of prediction should not be denied because model 4 at least can reach the prediction ability as strong as model 3, which includes a Cox' model in the survival part.

What's more, for some particular data sets that the hazard ratio of two individuals is changing over time, model 4 is highly likely to show its superiority.

In this part, two figures are plotted in section 5.5.1, helping to understand the behaviour of model 4; section 5.5.2 interprets the effect of eight clinical factors for the survival outcomes; in section 5.5.3, the empirical likelihood type loss function is discussed in a special case where the minimum point can be explicitly expressed.

5.5.1. Deepen into the new model

Model 4 ultimately estimates a distribution density function for each patient. Figure 5.3a shows two examples - the blue line represents the approximated density for a high-risk patient who dies at 270th day from being diagnosed, so a peak is observed at about 300th day; the yellow line stands for a low-risk patient's estimated density who survives to 3644th day and gets censored, hence the density is pretty flat and close to 0 before 5000th day, and has a peak at about 5400th day. The former can be verified to be a good approximation, while the latter is at least not unacceptable.

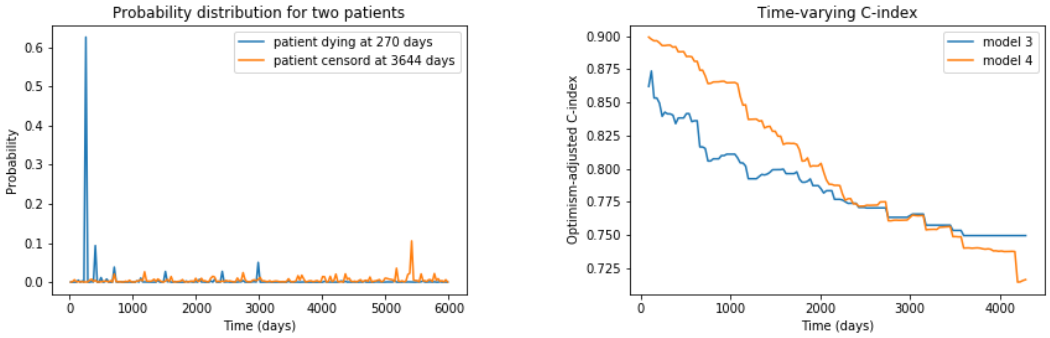
Furthermore, it is found that the estimated density is kind of rough, for the arising of multiple peaks (plotting the density from discretized probability mass will introduce certain roughness, but the main cause for multiple peaks lies in the unconcentrated estimated density). This inspires us to employ some smoothing techniques in loss function 5.1. An simply example is followed, for a non-censored individual who dies at interval l_3, its likelihood is a weighted average: $\frac{w \cdot p_2 + p_3 + w \cdot p_4}{1 + 2w}$, $w \in (0, 1)$. This can be a valuable further work to create a smoother density estimate and stabilize the prediction performance.

Figure 5.3b provides extra evidence for the over-performance of model 4. Now the time-varying C-index is evaluated on a more dense time set and two functions are approximated for the optimism-adjusted C-index of model 3 and 4. It can be seen that before 2400th, yellow-marked C-index of model 4 is consistently higher that blue-marked C-index of model 3. After an equivalent stage (from 2400th day to 3500th day), the blue line stably dominates the yellow line. If the area between the two lines is considered, the yellow one obviously has a larger positive area over the blue one, indicating the better prediction of model 4. Actually, the decreasing of the yellow line at the right side demonstrates a weakness of the new type of prediction - the predicted risk is defined as the estimated cumulative distribution function. When the time t is close to the life end, all of the patients' $\hat{F}(t)$ is nearly 1, giving rise to the low discrimination.

5.5.2. Model interpretation

Though deep networks offer tremendous success in survival prediction, it is hard to interpret the effect of some covariates. This massively prevents them from being widely used in medical research. Encouraged by Lee et al. (2019), the thesis defines the influence of a covariate Z_d (an element of the covariate vector $\boldsymbol{Z}$) at time t as the predicted risk difference between individuals with the maximum and the minimum value on Z_d:

$$\hat{F}(t|Z_d = z_{d,max}, \boldsymbol{Z}_{/d}) - \hat{F}(t|Z_d = z_{d,min}, \boldsymbol{Z}_{/d}), \tag{5.4}$$



(a) (b)

Figure 5.3.: (a) - estimated probability mass on each interval for two patients. (b) - the time-varying C-index (the apparent one introduced in section 4.2.3) evaluated at (90, 120, . . . , 4290) for the two models.

where $Z_d = z_{d,max}$ and $Z_d = z_{d,min}$ means an individual takes the maximum and the minimum value on Z_d, respectively, and $\boldsymbol{Z}_{/d}$ denotes the covariates in $\boldsymbol{Z}$ other than Z_d. If there are many subjects measured the maximum and the minimum value on Z_d, $\hat{F}(\cdot)$ can be an average over the predicted risks of those subjects; to avoid outliers, if there is only one subject measured the maximum and the minimum value on Z_d, $\hat{F}(\cdot)$ can be altered by an average over the predicted risks of subjects with biggest p or smallest p values on Z_d, where p is an integer, i.e. 5 as taken for table 5.3.

Each clinical factor's effect for the risk at 1800th day and 3600th day is listed in table 5.3. It is found that age is the most influential factor, followed by functional class and six-minute walk distance. These agree with the clinical commons. Also, males have a slight higher risk than that of females and the surprising thing is that the mean pulmonary artery pressure has little effect for survival outcomes. Actually, these points have been revealed in preliminary analysis in section 3.2. Moreover, the influence of RV1 and RV2 is quite messy, sometimes positive, sometimes negative. This may be due to their clinical meaning that the normal values should be in a certain interval and above or below the interval both have negative impact. Possibly, patients who die before 1800th day mostly have low RV1 and RV2 values, while the majority patients that die between 1800th to 3600th day are measured high RV1 and RV2 values. This analysis infers that at the beginning, low RV1 and RV2 are indicators for high risk, but after about 5 years, patients with high RV1 and RV2 values are in high risk. Apart from that, RV3 is a negative-effect factor.

	sex	age	FC	SMWD	MPAP	RV1	RV2	RV3
1800th day	0.0463	0.4159	0.1647	-0.1705	0.0069	-0.1559	-0.0608	-0.1759
3600th day	0.0420	0.4075	0.1353	-0.1356	-0.0100	0.2351	0.2142	-0.1071

Table 5.3.: Each clinical factor's effect for the risk at 1800th day and 3600th day. The positivity means a patient will stay at a high risk with large values of the covariates.

5.5.3. More about the empirical likelihood type loss function

The empirical likelihood type loss function is able to force the neural networks to learn the underlying distribution of the survival time. In a special case, the minimum point of the empirical likelihood (equation 5.1) can be analytically derived.

Proposition 16. *Suppose that $t_1 < t_2 < \cdots < t_k$ are the ordered death times, $k \leq N$, the number of samples. For completion, $t_0 = 0$. Denote the number of deaths at time t_j as d_j, and the discrete hazard rate at time t_j as μ_j, the number of individuals who are at risk at time t_j as n_j as well. Assume the support of the survival time X is $\mathcal{T} = (0, t_{max})$. If $\mathcal{T}$ is divided into $l_1 = (0, t_1], l_2 = (t_1, t_2], \ldots, l_{k+1} = (t_k, t_{max})$, the equation 5.1 is minimized by $\hat{p}_i = \hat{S}(t_i) - \hat{S}(t_{i-1})$, where $\hat{S}(t_i) = \prod_{j=1}^{i}(1 - \frac{d_j}{n_j})$, for $i = 1, 2, \ldots, k$; $\hat{p}_{k+1} = 1 - \hat{S}(t_k)$.*

Proof. Actually, this proposition is more like a corollary from Kaplan-Meier estimate (Appendix A.1). In this partition, the equation 5.1 is equivalent to the empirical likelihood in formula A.1. By the invariance of MLE, the results follows. □

By the proposition 16, the survival prediction is categorised into a classification problem again. Hence, the loss function can be replaced by a cross-entropy function:

$$L = -\sum_{n=1}^{N}\sum_{i=1}^{k+1}\Big[\hat{p}_i \log p_{ni}^{o} + (1 - \hat{p}_i)\log(1 - p_{ni}^{o})\Big], \tag{5.5}$$

where p_{ni}^{o} nth subject's output on $\hat{p}_i$.

6. Summary

This thesis follows the research of Bello et al. (2019) and explores further in two directions:

- proposing a novel approach for joining the high dimensional time-resolved cardiac motion features and comparably low dimensional clinical factors in a deep network to increase the survival prediction performance,
- incorporating a new type of survival prediction and forcing the deep network to learn the underlying probability distribution of the survival time.

For the first one, the clinical factors are added with latent code to magnify the effect of extra information and one more hidden layer is placed between the merged layer and the prediction layer to allow non-linear effect of clinical factors and the interaction between them and the latent code. This structure achieves about 3% higher prediction accuracy than that of the model in Bello et al. (2019) where only cardiac motion data are used. This improvement is statistically significant at the level of 0.05. The importance of the one more hidden layer is deeply analysed and it is concluded that the clinical factors behave like gaussian noise in the network processing from a linear perspective.

For the latter, the latent code and the clinical factors are fed to a softmax layer for probability distribution prediction, intermediated by a hidden layer to allow nonlinear relationship and optimised interaction. The new model also outperforms the optimal model in the first part by 3%, while only significant at the level of 0.1 due to the larger variation in survival prediction.

The effect of the clinical factors is interpreted in the last model (the model with new prediction). It is found that age is the most influential factor, followed by functional class and six- minute walk distance, and surprisingly, mean pulmonary artery pressure has little effect for survival outcomes. There is also no obvious survival discrepancy on different genders. Moreover, RV end-diastolic volume and RV end-systolic volume seem to have time-varying effect from the analysis and RV ejection fraction negatively impacts the patients' survival.

Some of the underlying theoretical ideas for survival neural networks are also discussed in the thesis. From the review in chapter 2, survival neural networks are mainly trained via minimizing cross-entropy error function or maximizing the likelihood. These are two different principles: the former is from a classification aspect and the latter originates from certain statistical models (e.g. Cox's model and the empirical model). In Biganzoli et al. (1998), they found an equivalence of the two principles in a special case when the output of the network was hazard rate (and with proper labelling). In the new prediction introduced in chapter 5, another case for the equivalence of the two principles is discovered (see proposition 16) - when the network learns the discretized probability

density of the survival time and the time line is partitioned with cut points at all the death time. This equivalence is theoretically important, in the sense that the model is optimal both at the two principles.

Due to the limited time, there are many further works worthy to do in the future:

- Dealing with missing values. It is really common in survival analysis that incomplete measurements are involved, e.g. the 8 clinical factors originally have missing values while imputed by MissForest (Stekhoven and Bühlmann, 2011). It is meaningful to propose new processing methods to handle it in high-dimensional setting, like for survival analysis with incomplete genomic data.
- Inspired by the new prediction in chapter 5, we can also assign more flexible task for the neural network, e.g. regarding the baseline hazard in Cox's model as a step-wise function and estimating the height in each step.
- A new time-varying prediction metric should be proposed. Although the inverse probability of censoring weighted C-index has fixed in the inconsistency problem, but it is not discriminative as t goes to the life end.
- Sample size is an issue in the thesis, resulting in large variation in prediction and probably unsuitable for outputting large amount of discretized probability density in chapter 5. It is worthwhile to run the models after acquiring more data.

Bibliography

O. O. Aalen. A linear regression model for the analysis of life times. *Statistics in medicine*, 8(8):907–925, 1989.

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2), 2015.

T. Ahmad, M. J. Pencina, P. J. Schulte, E. OâĂŹBrien, D. J. Whellan, I. L. Piña, D. W. Kitzman, K. L. Lee, C. M. OâĂŹConnor, and G. M. Felker. Clinical implications of chronic heart failure phenotypes defined by cluster analysis. *Journal of the American College of Cardiology*, 64(17):1765–1774, 2014.

P. K. Andersen and R. D. Gill. Cox's regression model for counting processes: a large sample study. *The annals of statistics*, pages 1100–1120, 1982.

P. B. Antolini, Laura and E. Biganzoli. A time dependent discrimination index for survival data. *Statistics in Medicine*, 2005.

G. A. Bello, T. J. Dawes, J. Duan, C. Biffi, A. de Marvao, L. S. Howard, J. S. R. Gibbs, M. R. Wilkins, S. A. Cook, D. Rueckert, et al. Deep-learning cardiac motion analysis for human survival prediction. *Nature machine intelligence*, 1(2):95, 2019.

C. Biffi, O. Oktay, G. Tarroni, W. Bai, A. De Marvao, G. Doumou, M. Rajchl, R. Bedair, S. Prasad, S. Cook, et al. Learning interpretable anatomical features through deep generative models: Application to cardiac remodeling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 464–471. Springer, 2018.

E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in medicine*, 17(10):1169–1186, 1998.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

P. P. Brahma, D. Wu, and Y. She. Why deep learning works: A manifold disentanglement perspective. *IEEE transactions on neural networks and learning systems*, 27(10):1997–2008, 2015.

A. E. Bryson and Y.-C. Ho. Optimization, estimation and control. *Ginn and Company*, 1969.

K. Burke and G. MacKenzie. Multi-parameter regression survival modeling: An alternative to proportional hazards. *Biometrics*, 73(2):678–686, 2017.

Z. Cai and Y. Sun. Local linear estimation for time-dependent coefficients in cox's regression models. *Scandinavian Journal of Statistics*, 30(1):93–111, 2003.

W.-T. Chang, S.-F. Weng, C.-H. Hsu, J.-Y. Shih, J.-J. Wang, C.-Y. Wu, and Z.-C. Chen. Prognostic factors in patients with pulmonary hypertensionâĂŤa nationwide cohort study. *Journal of the American Heart Association*, 5(9):e003579, 2016.

C.-L. Chi, W. N. Street, and W. H. Wolberg. Application of artificial neural network-based survival analysis on two breast cancer datasets. In *AMIA Annual Symposium Proceedings*, volume 2007, page 130. American Medical Informatics Association, 2007.

T. Ching, X. Zhu, and L. X. Garmire. Cox-nnet: an artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS computational biology*, 14(4):e1006076, 2018.

F. Chollet et al. Keras, 2015.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

M. De Laurentiis and P. M. Ravdin. A technique for using neural network analysis to perform survival analysis of censored data. *Cancer Letters*, 77(2-3):127–138, 1994.

B. Efron and T. Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

J. Fan, H. Lin, Y. Zhou, et al. Local partial-likelihood estimation for lifetime data. *The Annals of Statistics*, 34(1):290–325, 2006.

D. Faraggi and R. Simon. A neural network model for survival data. *Statistics in medicine*, 14(1):73–82, 1995.

L. Flavell. Pulmonary hypertension life expectancy. 2018. URL `https://pulmonaryhypertensionnews.com/pulmonary-hypertension-life-expectancy-2/`.

N. Galiè, M. Humbert, J.-L. Vachiery, S. Gibbs, I. Lang, A. Torbicki, G. Simonneau, A. Peacock, A. Vonk Noordegraaf, M. Beghetti, et al. 2015 esc/ers guidelines for the diagnosis and treatment of pulmonary hypertension: the joint task force for the diagnosis and treatment of pulmonary hypertension of the european society of cardiology (esc) and the european respiratory society (ers): endorsed by: Association for european paediatric and congenital cardiology (aepc), international society for heart and lung transplantation (ishlt). *European heart journal*, 37(1):67–119, 2015.

T. A. Gerds, M. W. Kattan, M. Schumacher, and C. Yu. Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring. *Statistics in Medicine*, 32(13):2173–2184, 2013.

D. Gopalan, M. Delcroix, and M. Held. Diagnosis of chronic thromboembolic pulmonary hypertension. *European Respiratory Review*, 26(143):160108, 2017.

F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati. Evaluating the yield of medical tests. *Jama*, 247(18):2543–2546, 1982.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

H. Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer, 1992.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, M. S. Lauer, et al. Random survival forests. *The annals of applied statistics*, 2(3):841–860, 2008.

J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*, volume 360. John Wiley & Sons, 2011.

E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):24, 2018.

J. Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2010.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

C. Lee, W. R. Zame, J. Yoon, and M. van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

C. Lee, J. Yoon, and M. Van Der Schaar. Dynamic-deephit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE Transactions on Biomedical Engineering*, 2019.

D. Y. Lin and L.-J. Wei. The robust inference for the cox proportional hazards model. *Journal of the American statistical Association*, 84(408):1074–1078, 1989.

N. Mantel. Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemother Rep*, 50:163–170, 1966.

T. Martinussen and T. H. Scheike. *Dynamic regression models for survival data*. Springer Science & Business Media, 2007.

M. A. Mazurowski. Radiogenomics: what it is and why it is important. *Journal of the American College of Radiology*, 12(8):862–866, 2015.

NHS. Overview - pulmonary hypertension. 2017. URL `https://www.nhs.uk/conditions/pulmonary-hypertension/`.

H. Nilsaz-Dezfouli, M. R. Abu-Bakar, J. Arasan, M. B. Adam, and M. A. Pourhoseingholi. Improving gastric cancer outcome prediction using single time-point artificial neural network models. *Cancer informatics*, 16:1176935116686062, 2017.

G. Savarese and L. H. Lund. Global public health burden of heart failure. *Cardiac failure review*, 3(1):7, 2017.

T. H. Scheike. The additive nonparametric and semiparametric aalen model as the rate function for a counting process. *Lifetime Data Analysis*, 8(3):247–262, 2002.

M. A. Simon and M. R. Pinsky. Right ventricular dysfunction and failure in chronic pressure overload. pages Article ID 568095, 7 pages, 2011. URL `https://doi.org/10.4061/2011/568095`.

O. Sitbon, M. Delcroix, E. Bergot, A. B. Boonstra, J. Granton, D. Langleben, P. E. Subías, N. Galiè, T. Pfister, J.-C. Lemarié, et al. Epitome-2: an open-label study assessing the transition to a new formulation of intravenous epoprostenol in patients with pulmonary arterial hypertension. *American heart journal*, 167(2):210–217, 2014.

G. C. Smith, S. R. Seaman, A. M. Wood, P. Royston, and I. R. White. Correcting for optimistic prediction in small data sets. *American journal of epidemiology*, 180(3): 318–324, 2014.

D. J. Stekhoven and P. Bühlmann. MissforestâĂŤnon-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

W. N. Street. A neural network model for prognostic prediction. In *ICML*, pages 540–546, 1998.

L. Tian, D. Zucker, and L. Wei. On the cox model with time-varying regression coefficients. *Journal of the American statistical Association*, 100(469):172–183, 2005.

T. C. Timmreck. *An introduction to epidemiology*. Jones & Bartlett Learning, 2002.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

D. M. Zucker, A. F. Karr, et al. Nonparametric survival analysis with time-dependent covariate effects: a penalized partial likelihood approach. *The Annals of Statistics*, 18 (1):329–353, 1990.

A. Related tools in survival analysis

In the overall thesis, Kaplan-Meier plots are employed for many times to give a visualization of survival functions. Log rank test is also mentioned to compare survival curves. Here, the mathematical descriptions for both are detailed for smoother reading.

A.1. Kaplan-Meier estimate

A distinguished milestone among survival models is the Kaplan-Meier estimator (Kaplan and Meier, 1958), which is in the nonparametric paradigm and assume that individuals will only be at risk at the death time of the collected data. Suppose we have n observations and the data is in the form of $\{t_i, \delta_i\}_{i=1}^n$, where t_i is ith subject's survival time and $\delta_i = 1$ indicates that ith subject is not censored. Let $t_1 < t_2 < \cdots < t_k$ be the ordered death times, with $k \leq n$ so that two deaths can occur at the same time. Denote the number of deaths at time t_j as d_j, and the discrete hazard rate at time t_j as μ_j, the number of individuals who are at risk at time t_j as n_j as well. Now we can write down the empirical likelihood of the data as followed:

$$L(\mu_1, \mu_2, \ldots, \mu_k) = \prod_{j=1}^{k} \mu_j^{d_j} (1 - \mu_j)^{n_j - d_j}, \tag{A.1}$$

which can be maximised by

$$\hat{\mu}_j = \frac{d_j}{n_j}, j = 1, 2, \ldots, k. \tag{A.2}$$

Then the Kaplan-Meier estimate for the survival function is

$$\hat{S}(t) = \prod_{j:t_j \leq t} (1 - \frac{d_j}{n_j}). \tag{A.3}$$

Kaplan-Meier estimate is also called product-limit estimate, in the sense that with denser and denser partition of the time axis, $\hat{S}(t) = \prod_{j:t_j \leq t}(1 - \frac{d_j}{n_j}) \approx \prod_{j:t_j \leq t}(1 - \mu_j) = S(t)$, where μ_j is the discrete hazard rate at time t_j.

A.2. Log rank test

It is of interest to determine whether populations in two or more stratifications have an equal survival curve. Log rank test, firstly proposed by Mantel (1966), provides a nonparametric approach to handle the problem. For simplicity but without the loss of generalization, the case only involving two strata is considered. Now the hypothesis problem is

$$H_0 : S_1(t) = S_2(t) \longleftrightarrow H_1 : S_1(t) \neq S_2(t),$$

where $S_1(\cdot)$ and $S_2(\cdot)$ are survival functions for strata 1 and 2, respectively. The notation in section A.1 is still followed. Let d_{ij} and n_{ij} be the corresponding numbers in strata $i(i = 1, 2)$. Under the null hypothesis and conditional on the number of deaths at time t_j, d_j, the distribution for d_{1j} and d_2j is a two dimensional hypergeometric distribution with probability function:

$$\prod_{i=1}^{2} \binom{n_{ij}}{d_{ij}} \binom{n_j}{d_j}^{-1}. \tag{A.4}$$

The conditional mean and variance of d_{ij} can be easily derived, with respectively,

$$e_{ij} = n_{ij} d_j n_j^{-1} \tag{A.5}$$

and

$$(W_j)_{ii} = n_{ij}(n_j - n_{ij}) d_j (n_j - d_j) n_j^{-2} (n_j - 1)^{-1}. \tag{A.6}$$

The conditional covariance of d_{1j} and d_{2j} can be expressed as

$$(W_j)_{12} = -n_{1j} n_{2j} d_j (n_j - d_j) n_j^{-2} (n_j - 1)^{-1}. \tag{A.7}$$

Hence, the test statistic can be constructed in the form of

$$w = \sum_{j=1}^{k} w_j, \quad w_j = (d_{1j} - e_{1j}, \ldots, d_{2j} - e_{2j})^T.$$

Particularly, w_j has conditional mean 0 and 2×2 variance matrix W_j. Under some independence assumption, the final standardized test statistic is

$$T = w^T W^{-1} w, W = \sum_{j=1}^{k} W_j. \tag{A.8}$$

Asymptotically, T follows a χ_1^2 distribution, and the null hypothesis will be rejected by large observation of T.

B. Code

The code for training and validation is similar in model 0, 1, 2, 3, 4 mentioned in the thesis. Here, only the code for model 4 is presented. See `https://github.com/kinshoo/4Dsurvival.git` for more code.

```
import os, sys, time, pickle, copy, h5py
import keras
from keras import backend as K
import tensorflow as tf

# ============================================================
# # GPU code for session activation
# os.environ["CUDA_VISIBLE_DEVICES"]="2"
# config = tf.ConfigProto()
# session = tf.Session(config=config)
# K.set_session(session)
# ============================================================

from keras.callbacks import EarlyStopping, Callback
from keras.models import Sequential, Model
from keras.layers import Input
from keras.layers.core import Dense, Dropout, Flatten
from keras.layers.convolutional import Convolution3D, MaxPooling3D,
    ZeroPadding3D
from keras.optimizers import Adam
from keras.regularizers import l1, l2, l1_l2
from keras.constraints import max_norm
from keras.models import load_model
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm
from sklearn.preprocessing import scale
import optunity
import lifelines.utils
from lifelines.utils import concordance_index
from lifelines import CoxPHFitter
from lifelines import KaplanMeierFitter


"----Loading in files [Training and Test]----"
#os.chdir(os.path.join(os.path.expanduser('~'),'/Users/apple/Desktop/Code
    /'))

#import input data: i_orig=list of patient IDs, y_orig=censoring status and
     survival times for patients, x_orig=input data for patients (i.e.
    motion descriptors [11,514-element vector])
with open('3dall_data.pkl', 'rb') as f: x3Ddat,ydat,idat,covdat,cmrvolsdat
     = pickle.load(f)
numcovs  = covdat.shape[1]+cmrvolsdat.shape[1]
xdat = np.concatenate((x3Ddat,covdat,cmrvolsdat), axis=-1)
inpshape = x3Ddat.shape[1]

#longest time to survive
T_max = 6000

def prepare_data(x, e, t, c):
return (x.astype("float32"), e.astype("int32"), t.astype("float32"), c.
```

```
    astype("float32"))


def sort4minibatches(xvals, evals, tvals, cvals, batchsize):
ntot = len(xvals)
indices = np.arange(ntot)
np.random.shuffle(indices)
start_idx=0
esall = []
for end_idx in list(range(batchsize, batchsize*(ntot//batchsize)+1,
    batchsize))+[ntot]:
excerpt = indices[start_idx:end_idx]
sort_idx = np.argsort(tvals[excerpt])[::-1]
es = excerpt[sort_idx]
esall += list(es)
start_idx = end_idx
return (xvals[esall], evals[esall], tvals[esall], cvals[esall], esall)


def normz(x):
return scale(x, axis=0, with_mean=True, with_std=True)


#Define Cox PH partial likelihood function loss.
#Arguments: E (censoring status), risk (risk [log hazard ratio] predicted
    by network) for batch of input subjects
def _negative_log_likelihood(Ind, Prob):
return -K.sum(K.log(K.sum(Ind * Prob, axis = 1)))


#compute the indicator for probabilities that will be used in likelihood
    calculation
def get_Prob(e, tm, nbInt):
e = e.reshape(-1)
tm = tm.reshape(-1)
gap = T_max / (nbInt - 1)
interval = np.ceil(tm / gap)

indicator = np.zeros((len(e), nbInt))
for i in np.array(range(0, len(e))):
if(e[i] == 1):
tmp = np.zeros((nbInt, ))
tmp[(interval[i].astype("int") - 1)] = 1
indicator[i] = tmp
else:
tmp = np.ones((nbInt, ))
tmp[:(interval[i].astype("int"))] = np.zeros(((interval[i].astype("int32"))
    , ))
indicator[i] = tmp

return indicator


#use Kaplan-Meier estimate for censoring probabilities
def CensoringProb(Y, T):
T = T.reshape([-1]) # (N,) - np array
Y = Y.reshape([-1]) # (N,) - np array

kmf = KaplanMeierFitter()
kmf.fit(T, event_observed=(Y==0).astype(int))  # censoring prob = survival
    probability of event "censoring"
G = np.asarray(kmf.survival_function_.reset_index()).transpose()
G[1, G[1, :] == 0] = G[1, G[1, :] != 0][-1]  #fill 0 with smallest
    probability (to prevent nan values)

return G

```

```
109
110 #compute weighted C-index
111 def weighted_c_index(Y_train, T_train, Prediction, Y_test, T_test, Time):
112 G = CensoringProb(Y_train, T_train)
113
114 N = len(Prediction)
115 A = np.zeros((N,N))
116 Q = np.zeros((N,N))
117 N_t = np.zeros((N,N))
118 Num = 0
119 Den = 0
120 for i in range(N):
121 tmp_idx = np.where(G[0,:] >= T_test[i])[0] ##index of subjects whose time
        is >= T_test[i]
122
123 if len(tmp_idx) == 0:
124 W = (1. / G[1, -1])**2 ##G(T_i-)
125 else:
126 W = (1. / G[1, tmp_idx[0]])**2 ##G(T_i)
127
128 A[i, np.where(T_test[i] < T_test)] = 1. * W ## I(T_i<T_j) * W
129 Q[i, np.where(Prediction[i] > Prediction)] = 1. ## P_i > P_j
130
131 if (T_test[i]<=Time and Y_test[i]==1):
132 N_t[i,:] = 1.
133
134 Num = np.sum(((A) * N_t) * Q) ##sum all the elements in the array
135 Den = np.sum((A) * N_t)
136
137 if Num == 0 and Den == 0:
138 result = -1 # not able to compute c-index!
139 else:
140 result = float(Num/Den)
141
142 return result
143
144
145 "------define architecture and fitting ------"
146 def modfit(xtr, ytr, alpha, dro, units1, units2, units3, l1r, lr, batchsize
      , numepochs):
147 X_tr, E_tr, TM_tr, C_tr = prepare_data(xtr[:,:-numcovs], ytr[:,0,np.newaxis
      ], ytr[:,1], xtr[:,-numcovs:])
148
149 #Arrange data into minibatches (based on specified batch size), and within
      each minibatch, sort in descending order of survival/censoring time (
      see explanation of Cox PH loss function definition)
150 X_tr, E_tr, TM_tr, C_tr, _ = sort4minibatches(normz(X_tr), E_tr, TM_tr,
      normz(C_tr), batchsize)
151
152 #before defining network architecture, clear current computation graph (if
      one exists), and specify input dimensionality
153 K.clear_session()
154
155 #Specify Model Architecture
156 inputvec= Input(shape=(inpshape,), name = 'inputvec')
157 inputconv = Input(shape=(numcovs,), name = 'inputconv')
158 x       = Dropout(dro, input_shape=(inpshape,))(inputvec)
159 x       = Dense(units=int(units1), activation='relu', activity_regularizer=
      l1(10**l1r))(x)
160 encoded = Dense(units=int(units2), activation='relu', name='encoded')(x)
161
162 merge_layer = keras.layers.concatenate([encoded, inputconv])
163 riskpred= Dense(units=int(units3), activation='softmax', name='
      predicted_risk')(merge_layer)
164 z       = Dense(units=int(units1), activation='relu')(encoded)
165 decoded = Dense(units=inpshape, activation='linear', name='decoded')(z)
166
```

```
#Define model inputs (11,514-element motion descriptor vector and 8 clinical factors) and outputs (reconstructed input and predicted risk)
model = Model(inputs=[inputvec, inputconv], outputs=[decoded,riskpred])
model.summary()

#Model compilation
optimdef = Adam(lr = lr)

model.compile(loss=[keras.losses.mean_squared_error, _negative_log_likelihood], loss_weights=[alpha,1-alpha], optimizer= optimdef, metrics={'decoded':keras.metrics.mean_squared_error})

##Indicator for the probability
Ind_prob = get_Prob(E_tr, TM_tr, int(units3))

#Run model
mlog = model.fit([X_tr, C_tr], [X_tr, Ind_prob], batch_size=batchsize, epochs=numepochs, shuffle=False, verbose=1)
del mlog

return model


#prediction accuracy
def chimpred_new(m_MLP, y_train, x_test, y_test, eval_t = None):
e_train = y_train[:, 0]
tm_train = y_train[:, 1]
e_test = y_test[:, 0]
tm_test = y_test[:, 1]

Xdat_test = normz(x_test[:,:-numcovs])
Cdat_test = normz(x_test[:,-numcovs:])

preds = m_MLP.predict([Xdat_test, Cdat_test], batch_size=1)[1]
nbInt = preds.shape[1]

if eval_t is None:
eval_t = [int(np.percentile(np.array([tm_train, tm_test]).reshape(-1), 25)) , int(np.percentile(np.array([tm_train, tm_test]).reshape(-1), 50)), int(np.percentile(np.array([tm_train, tm_test]).reshape(-1), 75))]

result = np.zeros(len(eval_t))

for t, t_time in enumerate(eval_t):
if t_time >= T_max:
print('ERROR: evaluation horizon is out of range')
result[t] = -1;
else:
tm_train = tm_train.reshape(-1)
gap = T_max / (nbInt - 1)
interval = np.ceil(t_time / gap)
risk = np.sum(preds[:, :interval.astype("int")], axis = 1)
result[t] = weighted_c_index(e_train, tm_train, risk, e_test, tm_test, t_time)

C = result.mean()

return preds, C


#search optimal hyper-parameters
def hypersearch_DL(x_data, y_data, method, nfolds, nevals, batch_size, num_epochs, alpha_range, dro_range, units1_range, units2_range, units3_range, l1r_range):
@optunity.cross_validated(x=x_data, y=y_data, num_folds=nfolds)
def modelrun(x_train, y_train, x_test, y_test, alpha, dro, units1, units2, units3, l1r):
```

```
224 mod_MLP = modfit(xtr=x_train, ytr=y_train, alpha = alpha, dro = dro, units1
        = units1, units2 = units2, units3 = units3, l1r = l1r, lr =
        10**(-4.83), batchsize = batch_size, numepochs = num_epochs)
225 _, cv_C = chimpred_new(mod_MLP, y_train, x_test, y_test, eval_t = None)
226 return cv_C
227 optimal_pars, searchlog, _ = optunity.maximize(modelrun, num_evals=nevals,
        solver_name=method, alpha=alpha_range, l1r=l1r_range, units1 =
        units1_range, units2= units2_range, units3 = units3_range, dro=
        dro_range)
228 print('Optimal hyperparameters : ' + str(optimal_pars))
229 print('Cross-validated C after tuning: %1.3f' % searchlog.optimum)
230 return optimal_pars, searchlog
231
232
233
234 "-------Main-------"
235 preds_full = []
236 preds_bootfull = []
237 preds_boot = []
238 inds_inbag = []
239 Cb_opts  = []
240
241 #STEP 1
242 #(1a) find optimal hyperparameters
243 opars, osummary = hypersearch_DL(x_data=xdat, y_data=ydat, method='particle
        swarm', nfolds=10, nevals=50, batch_size=16, num_epochs=100,
        units3_range=[5,100], l1r_range=[-7,-4], dro_range=[.1,.9],
        units1_range=[75,250], units2_range=[5,20], alpha_range=[0.3,0.7])
244
245 #(1b) using optimal hyperparameters, train a model on full sample
246 omMLP = modfit(xtr = xdat, ytr = ydat, alpha = opars['alpha'], dro = opars[
        'dro'], units1 = opars['units1'], units2 = opars['units2'], units3 =
        opars['units3'], lr=10**(-4.83), l1r = opars['l1r'], batchsize=16,
        numepochs=100)
247
248 #(1c) Compute Harrell's Concordance index
249 predfull, C_app = chimpred_new(omMLP, ydat, xdat, ydat, eval_t = None)
250 preds_full.append(predfull)
251 print('Apparent concordance index = {0:.4f}'.format(C_app))
252 print('done with app step')
253
254
255 #BOOTSTRAP SAMPLING
256
257 #define useful variables
258 nsmp = len(xdat)
259 rowids = [_ for _ in range(nsmp)]
260 B = 500
261 for b in range(B):
262 print('\n-----------------------------------------')
263 print('Current bootstrap sample:', b, 'of', B-1)
264 print('-----------------------------------------')
265
266 #STEP 2: Generate a bootstrap sample by doing n random selections with
        replacement (where n is the sample size)
267 b_inds = np.random.choice(rowids, size=nsmp, replace=True)
268 xboot = xdat[b_inds]
269 yboot = ydat[b_inds]
270
271 #(2a) find optimal hyperparameters
272 bpars = opars
273
274 #(2b) using optimal hyperparameters, train a model on bootstrap sample
275 bmMLP = modfit(xtr=xboot, ytr=yboot, alpha = opars['alpha'], dro = opars['
        dro'], units1 = opars['units1'], units2 = opars['units2'], units3 =
        opars['units3'], lr=10**(-4.83), l1r = opars['l1r'], batchsize=16,
        numepochs=100)
```

```
276
277 #(2c[i]) Using bootstrap-trained model, compute predictions on bootstrap
        sample. Evaluate accuracy of predictions (Harrell's Concordance index)
278 predboot, Cb_boot      = chimpred_new(bmMLP, yboot, xboot, yboot, eval_t =
        None)
279
280 #(2c[ii]) Using bootstrap-trained model, compute predictions on FULL sample
        .     Evaluate accuracy of predictions (Harrell's Concordance index)
281 predbootfull, Cb_full = chimpred_new(bmMLP, yboot, xdat, ydat, eval_t =
        None)
282
283 #STEP 3: Compute optimism for bth bootstrap sample, as difference between
        results from 2c[i] and 2c[ii]
284 Cb_opt = Cb_boot - Cb_full
285 #store data on current bootstrap sample (predictions, C-indices)
286 preds_boot.append(predboot)
287 preds_bootfull.append(predbootfull)
288 inds_inbag.append(b_inds)
289 Cb_opts.append(Cb_opt)
290 del bmMLP
291
292
293 #STEP 5
294 #Compute bootstrap-estimated optimism (mean of optimism estimates across
        the B bootstrap samples)
295 C_opt = np.mean(Cb_opts)
296
297 #Adjust apparent C using bootstrap-estimated optimism
298 C_adj = C_app - C_opt
299
300 #compute confidence intervals for optimism-adjusted C
301 C_opt_95confint = np.percentile([C_app - o for o in Cb_opts], q=[2.5,
        97.5])
302
303 #save summary of results
304 print('\n\n=======SUMMARY - TRAINING & VALIDATION FOR DEEP LEARNING MODEL
        =======\n')
305 print('Apparent concordance index = {0:.4f}\n'.format(C_app))
306 print('Optimism bootstrap estimate = {0:.4f}\n'.format(C_opt))
307 print('Optimism-adjusted concordance index = {0:.4f}, and 95% CI = {1}\n'.
        format(C_adj, C_opt_95confint))
308
309
310 "------Save the results------"
311 with open('my3dall_new_output.pkl', 'wb') as f: pickle.dump([idat, ydat,
        preds_full, preds_boot, preds_bootfull, inds_inbag, C_app, Cb_opts,
        opars],f)
```

Listing B.1: Training and validation for model 4