Highlights

Case-Base Neural Network: survival analysis with time-varying, higher-order interactions

Jesse Islam, Maxime Turgeon, Robert Sladek, Sahir Bhatnagar

- Case-Base Neural Networks (CBNNs) provide a neural network methodology for survival analysis that leverages a statistical sampling technique, removing the need for complex implementations while providing data-driven flexibility in full hazard function estimation.
- CBNNs provide an advantage over competing neural network survival models, as it naturally accounts for censoring, uses binary cross-entropy as the loss function and predicts a smooth in-time risk function.
- CBNNs outperform the competing models in a simulation, two studies and shows similar performance in the third.
- Our results highlight the benefit of combining case-base sampling with deep learning to provide a simple and flexible modeling framework for data-driven, time-varying interaction modeling of single event survival outcomes.

Case-Base Neural Network: survival analysis with time-varying, higher-order interactions

Jesse Islam[a], Maxime Turgeon[b], Robert Sladek[a,c], Sahir Bhatnagar[d]

[a] *McGill University Department of Quantitative Life Sciences, 805 rue Sherbrooke O, Montréal, H3A 0B9, Quebec, Canada*
[b] *University of Manitoba Department of Statistics, 50 Sifton Rd, Winnipeg, R3T2N2, Manitoba, Canada*
[c] *McGill University Department of Human Genetics, 805 rue Sherbrooke O, Montréal, H3A 0B9, Quebec, Canada*
[d] *McGill University Department of Biostatistics, 805 rue Sherbrooke O, Montréal, H3A 0B9, Quebec, Canada*

Abstract

Neural network-based survival methods can model data-driven covariate interactions. While these methods can provide better predictive performance than regression-based approaches, not all can model time-varying interactions and complex baseline hazards. To address this, we propose Case-Base Neural Networks (CBNNs) as a new approach that combines the case-base sampling framework with flexible neural network architectures. Using a novel sampling scheme and data augmentation to naturally account for censoring, we construct a feed-forward neural network that may take time as an input. CBNNs predict the probability of an event occurring at a given moment to estimate the hazard function. We compare the performance of CBNNs to regression and neural network-based survival methods in a simulation and three case studies using two time-dependent metrics. First, we examine performance on a simulation involving a complex baseline hazard and time-varying interactions to assess all methods, with CBNN outperforming competitors. Then, we apply all methods to three real data applications, with CBNNs outperforming the competing models in two studies and showing similar performance in the third. Our results highlight the benefit of combining case-base sampling with deep learning to provide a simple and flexible modeling framework for data-driven, time-varying interaction modeling of single event survival outcomes. An R package is available at https://github.com/Jesse-Islam/cbnn.

Keywords: survival analysis,, machine learning, case-base, neural network

1. Introduction

A common assumption in survival analysis is that the risk of the event of interest does not vary with time (i.e. proportional hazards) [1]. This simplifying assumption results in using Cox proportional hazards models more often than smooth-in-time, accelerated failure time (AFT) models [1]. This preference causes analyses to be based on hazard ratios and relative risks rather than on survival curves and absolute risks [1]. The proportional hazards assumption in a Cox model may be incorrect in studies where the disease pathogenesis may change over time [2]. For example, studies of breast cancer incidence have shown that the risk of developing a tumor involves a time-varying interaction with tumor size [2]. These time-varying covariates can be incorporated easily into AFT models [3]. However, these regression-based models require prior knowledge of potential time-varying interactions and their quantitative effects.

Neural networks provide a data-driven approach to approximating interaction terms that benefit prediction. For example, DeepSurv is a neural network-based proportional hazards model that implements the Cox partial log-likelihood as a custom loss function [4]. This results in a stepwise absolute risk curve that cannot accommodate time-varying interactions. DeepSurv performs better than Cox proportional hazard models in the Rotterdam and German Breast Cancer Study Group and The Molecular Taxonomy of Breast Cancer International Consortium datasets [4]. It has also been used to develop personalized prediction models for cancers in a number of studies [5] [6] [7]. Previous work suggests we may change the loss function to address non-proportional hazards [8]. In contrast, DeepHit assumes an inverse Gaussian distribution as the baseline hazard [9] and specifies each follow-up time of interest in the model. DeepHit directly estimates survival curves, rather than deriving a hazard function [9]. On real datasets, DeepHit outperformed DeepSurv [9]. We note that DeepSurv and DeepHit both use custom loss functions [4] [9]. In addition, DeepHit introduces a hyperparameter weighing its two loss functions (negative log-likelihood and ranking losses) [9]. Compared to these neural network methods, regression-based approaches require prior specification of all interaction terms, which makes it challenging to model covariate effects that change with time. The current neural network models provide flexibility at the cost of clarity, while regression models provide clarity at the cost of flexibility.

We propose Case-Base Neural Networks (CBNNs) as a method for single event survival analysis that models time-varying interactions and a flexible baseline hazard. CBNNs use the case-base sampling technique, which allows probabilistic models to predict survival outcomes [1]. After case-base sampling, we implement a model using common neural network components that uses transformations of time as a feature (covariate) to specify different baseline hazards. For example, by including splines of time as covariates, case-base with logistic regression (CBLR) can approximate the Royston-Parmar flexible baseline hazard [3] [1]. However, this still requires explicit specification of time-varying interactions. CBNN can model both without extra tuning parameters.

In this paper, we describe the case-base sampling method and compare its properties to neural network models, along with our hyperparameter selection procedure, metrics and software implementation. Next, we explore the performance of CBNN, DeepSurv, DeepHit, Cox regression and CBLR on simulated data and describe their performance in three case studies. Finally, we explore the implications of our results and contextualize them within neural network survival analysis in a single event setting.

2. Case-base neural network, metrics, hyperparameters and software

Case-base sampling is an alternative framework for survival analysis [1], which converts the total survival time into discrete person-moments. In this section, we detail how neural networks explicitly incorporate time as a feature while adjusting for the sampling bias in case-base sampled data and describe the metrics and hyperparameter selection procedure used to compare CBNN with other software packages. An R package to use CBNN is available at `https://github.com/Jesse-Islam/cbnn`. The entire code base to reproduce the figures and empirical results in this paper is available at `https://github.com/Jesse-Islam/cbnnManuscript`.

2.1. Case-base sampling

To implement case-base sampling, we divide the total survival time for each individual into discrete person-specific moments (person-moments) and treat each person-moment as a sample. This creates a *base series* of *person-moments* where an event does not occur. This *base series* complements the *case series*, which contains all person-moments at which the event of interest occurs.

For each person-moment sampled, let X_i be the corresponding covariate profile $(x_{i1}, x_{i2}, ..., x_{ip})$, T_i be the time of the person-moment and Y_i be the indicator variable for whether the event of interest occurred at time T_i. We estimate the hazard function $h(t \mid X_i)$ using the sampled person-moments. Recall that $h(t \mid X_i)$ is the instantaneous risk of experiencing the event at time t for a given set of covariates X_i, assuming $T_i \geq t$.

Now, let b be the (user-defined) size of the *base series* and let B be the sum of all follow-up times for the individuals in the study. Let c be the number of events in the *case series*. A reasonable concern is determining how large b should be relative to c, since the size of b influences how much information is lost in the sampling process [1]. The relative information when comparing two averages is measured by $\frac{cb}{c+b} = \left[\left(\frac{1}{c} + \frac{1}{b}\right)\right]$, where b is the sample size of the *base series* and c is the sample size of the *case series* [1] [10]. If $b = 100c$, then we expect learned weights to be at most one percent higher than if the entire study base B was used, as they are proportional to $\frac{1}{c} + \frac{1}{100c}$ rather than $\frac{1}{c} + \frac{1}{\infty c}$ [1] [10].

If we sample the *base series* uniformly across the study base, then the hazard function of the sampling process is equal to b/B. Therefore, we have the following equality [11] [For a rigorous treatment, see Saarela & Hanley (2015) section 3].:

$$\frac{P\left(Y_i = 1 \mid X_i, T_i\right)}{P\left(Y_i = 0 \mid X_i, T_i\right)} = \frac{h\left(T_i \mid X_i\right)}{b/B}. \tag{1}$$

The odds of a person-moment being in the *case series* is the ratio of the hazard $h(T_i \mid X_i)$ and the uniform rate b/B. Using (1), we can see how the log-hazard function can be estimated from the log-odds arising from case-base sampling:

$$\log\left(h\left(t \mid X_i\right)\right) = \log\left(\frac{P\left(Y_i = 1 \mid X_i, t\right)}{P\left(Y_i = 0 \mid X_i, t\right)}\right) + \log\left(\frac{b}{B}\right). \tag{2}$$

To estimate the correct hazard function, we adjust for the bias introduced when sampling a fraction of the study base $(\frac{b}{B})$ by adding the term $\log\left(\frac{B}{b}\right)$ to offset $\log\left(\frac{b}{B}\right)$ during the fitting process.

2.2. Neural networks to model the hazard function

After case-base sampling, we pass all features, including time, into any user-defined feed-forward component (Figure 1). Then, we add an offset term and pass the output through a sigmoid activation function (Figure 1). Since we are interested in predicting the odds of an event occurring, the sigmoid activation function is ideal as it is the inverse of the odds and can be used to calculate the hazard. The general form for the neural network using CBNN is:

$$P\left(Y=1|X,T\right)=\text{sigmoid}\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right),$$

where T is a random variable representing the event time, X is the random variable for a covariate profile, $f_\theta(X,T)$ represents any feed-forward neural network architecture, $\log\left(\frac{B}{b}\right)$ is the offset term to adjust for the bias ($\log\left(\frac{b}{B}\right)$) set by case-base sampling, θ is the set of parameters learned by the neural network and $\text{sigmoid}(x)=\frac{1}{1+e^{-x}}$. By approximating a higher-order polynomial of time using a neural network, the baseline hazard specification is now data-driven, while user-defined hyperparameters such as regularization, number of layers and nodes control the flexibility of the hazard function.

The following derivation shows how our probability estimate is converted to odds:

$$\begin{aligned}
\log\left(h(t\mid X)\right) &= \log\left(\frac{\text{sigmoid}\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{1-\text{sigmoid}\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}\right)+\log\left(\frac{b}{B}\right)\\
&= \log\left(\frac{\frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}{1-\frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}\right)+\log\left(\frac{b}{B}\right)\\
&= \log\left(\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)\right)+\log\left(\frac{b}{B}\right)\\
&= f_\theta(X,T)+\log\left(\frac{B}{b}\right)+\log\left(\frac{b}{B}\right)\\
&= f_\theta(X,T).
\end{aligned}$$

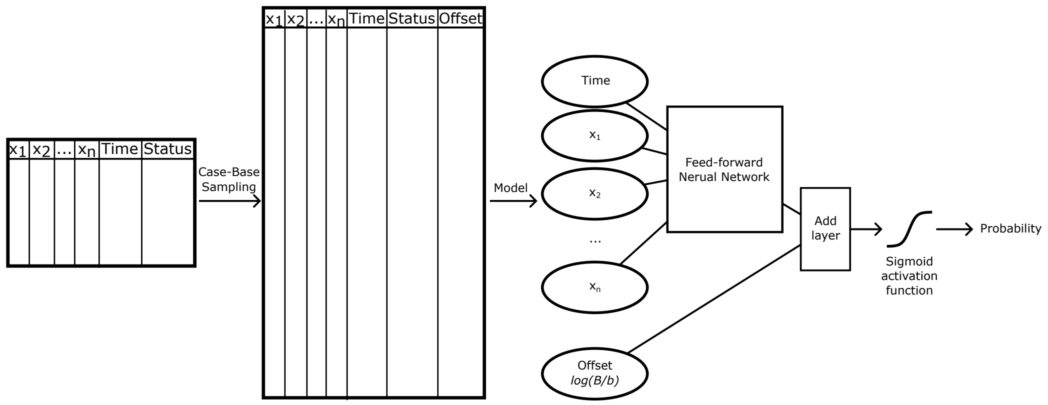

Figure 1: Methodological steps involved in CBNN. The first step, case-base sampling, is completed before training begins. Then, we pass this sampled data through a feed-forward neural network and add an offset to adjust for the bias inherent in case-base sampling and apply a sigmoid activation function to create a probability. Once the neural network model completes its training, we can convert the probability output to a hazard for the survival outcome of interest.

We use binary cross-entropy as our loss function [12]:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{f}_\theta(x_i, t_i)) + (1 - y_i) \cdot \log(1 - \hat{f}_\theta(x_i, t_i)),$$

where $\hat{f}_\theta(x_i, t_i)$ is our estimate for a given covariate profile and time, y_i is our target value specifying whether an event occurred and N represents the number of individuals in our training set.

Backpropagation with an appropriate minimization algorithm (such as Adam, RMSPropagation, stochastic gradient descent) is used to optimize the parameters in the model [12]. For our analysis, we use Adam as implemented in Keras [12]. While the size of the *case series* is fixed as the number of events; the size of the *base series* is not restricted. We use a ratio of 100:1 *base series* to *case series* [1]. After fitting our model, we convert the output to a hazard. To use CBNN for predictions, we manually set the offset term $\left(\log\left(\frac{B}{b}\right)\right)$ to 0 in the new data as we already account for the sampling bias during the fitting process.

Since we are directly modeling the hazard, we can readily estimate the risk function (F) at time t for a covariate profile X,

$$F\,(t \mid X) = 1 - \exp\left(-\int_0^t h(u|X)\,\mathrm{d}u\right). \tag{3}$$

We use a finite Riemann sum [13] to approximate the integral in (3).

2.3. Performance metrics

To choose our method-specific hyperparameters, we use the Integrated Brier Score (IBS) [14], which is based on the Brier Score (BS) and provides a summarized assessment of performance for each model. We assess the performance of models on a held out dataset using two metrics: 1) the Index of Prediction Accuracy (IPA) [15]; and 2) the Inverse probability censoring weights-adjusted time-dependent area under the receiver operating characteristic curve (AUC_{IPCW}) [16]. The IPA score is a metric for both discrimination and calibration, while the AUC_{IPCW} provides a metric for discrimination only.

2.3.1. Brier Score (BS)

The BS [14] is defined as

$$BS(t) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\left(1-\widehat{F}(t \mid X_i)\right)^2 \cdot I(T_i \le t, \delta_i = 1)}{\widehat{G}(T_i)} + \frac{\widehat{F}(t \mid X_i)^2 \cdot I(T_i > t)}{\widehat{G}(t)}\right), \tag{4}$$

where $\delta_i = 1$ shows individuals who have experienced the event, N represents the number of samples in our dataset over which we calculate $BS(t)$, T_i is the survival or censoring time of an individual, $\widehat{G}(t) = P[c > t]$ is a non-parametric estimate of the censoring distribution and c is censoring time. The BS provides a score that accounts for the information loss because of censoring. Once we fix our t of interest, the individuals in the dataset can be divided into three groups. Individuals who experienced the event before t are present in the first term of the equation. The second term of the equation includes individuals who experience the event or are censored after t. Those censored before t (the remaining individuals) are accounted for by the IPCW adjustment ($G(\cdot)$) present on both terms.

2.3.2. Integrated Brier Score (IBS)

The Integrated Brier Score (IBS) is a function of the BS (4) [14], which is defined as

$$IBS(t) = \int_t^{t_{max}} BS(t)dw(t),$$

where $w(t) = \frac{t}{t_{max}}$ and t_{max} is the upper bound for the survival times of interest. As the IBS is calculated for a range of follow-up-times, it is a useful metric to track overall performance across all of follow-up time during cross-validation. We use the IPA score to get an estimate of performance at each follow-up time for our studies.

2.3.3. Index of prediction accuracy (IPA)

The IPA is a function of time, based on the BS. The IPA score is given by

$$\text{IPA}(t) = 1 - \frac{BS_{model}(t)}{BS_{null}(t)},$$

where $(BS_{model}(t))$ represents the BS over time (t) for the model of interest and $(BS_{null}(t))$ represents the BS if we use an unadjusted Kaplan-Meier (KM) curve as the prediction for all observations [15]. The IPA score has an upper bound of one, where positive values show an increase in performance over the null model and negative values show that the null model performs better. As an extension of BS, the IPA score assesses both model calibration and discrimination and shows how performance changes over follow-up time [14] [15].

2.3.4. Inverse probability censoring weights-adjusted time-dependent area under the receiver operating characteristic curve AUC_{IPCW}

The IPCW-adjusted AUC (AUC_{IPCW}) is a time-dependent metric that considers censoring [16]. For a given follow-up time of interest,

$$AUC_{IPCW}(t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} I_{\widehat{F}(t|X_i)>\widehat{F}(t|X_j)} \cdot I_{T_i \le t, \delta_i=1} \cdot \left(1 - I_{T_j \le t, \delta_j=1}\right) \cdot W_i(t) \cdot W_j(t)}{\sum_{i=1}^{n}\sum_{j=1}^{n} I_{T_i \le t, \delta_i=1} \cdot \left(1 - I_{T_j \le t, \delta_j=1}\right) \cdot W_i(t) \cdot W_j(t)},$$

where

$$W_i(t) = \frac{I_{T_i \le t, \delta_i=1}}{\widehat{G}(T_i)} + \frac{I_{T_i > t}}{\widehat{G}(t)}.$$

The AUC_{IPCW} measures discrimination [16]. By examining both IPA and AUC_{IPCW}, we can better understand whether a model performs better in terms of calibration or discrimination.

2.4. Hyperparameter selection

Neural networks require external parameters that constrain the learning process (hyperparameters). We apply the following hyperparameter optimization procedure to each method for each study. First, we fix a test set with 15% of the data which we keep aside during hyperparameter selection. To determine the best hyperparameters for each neural network method, we use a three-fold cross-validated grid search [12] on the remaining data (85% training, 15% validation) for the following range of hyperparameters:

$$Search\ space: \begin{cases} Learning\ rate \sim \{0.001, 0.01\} \\ Dropout \sim \{0.01, 0.05, 0.1\} \\ First\ layer\ nodes \sim \{50, 75, 100\} \\ Second\ layer\ nodes \sim \{10, 25, 50\} \\ Number\ of\ batches \sim \{100, 500\} \\ Activation\ function \sim \{ReLU, Linear\} \\ \alpha \sim \{0, 0.5, 1\}.\ (DeepHit\ only) \end{cases}$$

DeepHit uses α as a hyperparameter, while DeepSurv and CBNN do not. We track IBS on the validation set for each hyperparameter combination, choosing the combination with the lowest score for each method (Table 1).

Table 1: Hyperparameters selected after three-fold cross-validated grid search along with the average IBS for each neural network model in the complex simulation (A); multiple myeloma (MM) case study (B); free light chain (FLC) case study (C); and prostate cancer (Prostate) case study (D).

A: Complex

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.001	0.001
Dropout	0.01	0.01	0.05
First layer nodes	75	50	50
Second layer nodes	50	50	50
Number of batches	100	500	100
Activation function	relu	relu	linear
α	-	-	1
IBS	0.06744259	0.07529775	0.08821864

B: MM

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.01	0.01
Dropout	0.1	0.05	0.01
First layer nodes	50	100	100
Second layer nodes	50	25	25
Number of batches	500	100	100
Activation function	relu	relu	relu
α	-	-	0
IBS	0.09414844	0.09912649	0.1097949

C: FLC

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.001	0.01
Dropout	0.05	0.05	0.1
First layer nodes	50	100	50
Second layer nodes	10	10	10
Number of batches	100	100	100
Activation function	relu	relu	relu
α	-	-	1
IBS	0.09903534	0.09900328	0.09979871

D: Prostate

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.01	0.01
Dropout	0.01	0.1	0.05
First layer nodes	100	75	75
Second layer nodes	25	25	25
Number of batches	100	100	500
Activation function	relu	relu	linear
α	-	-	1
IBS	0.07618916	0.07631809	0.07634624

2.5. Software implementation

R [17] and Python [18] are used to evaluate methods from both languages. We fit the Cox model using the **survival** package [19] and the CBLR model using the **casebase** package [20]. Both DeepSurv and DeepHit are fit using **pyCox** [9]. We made the components of CBNN using the **casebase** package [20] for the sampling step and the **keras** package [21] for our neural network architecture. We use the **simsurv** package [22] for our simulation studies and **flexsurv** [23] to fit a flexible baseline hazard using splines for our complex simulation. The **riskRegression** package [24] is used to get the IPA and AUC_{IPCW}. We modify the **riskRegression** package to be used with any user supplied risk function F. We use the **reticulate** package [25] to run both R based methods and Python based methods on the same seed.

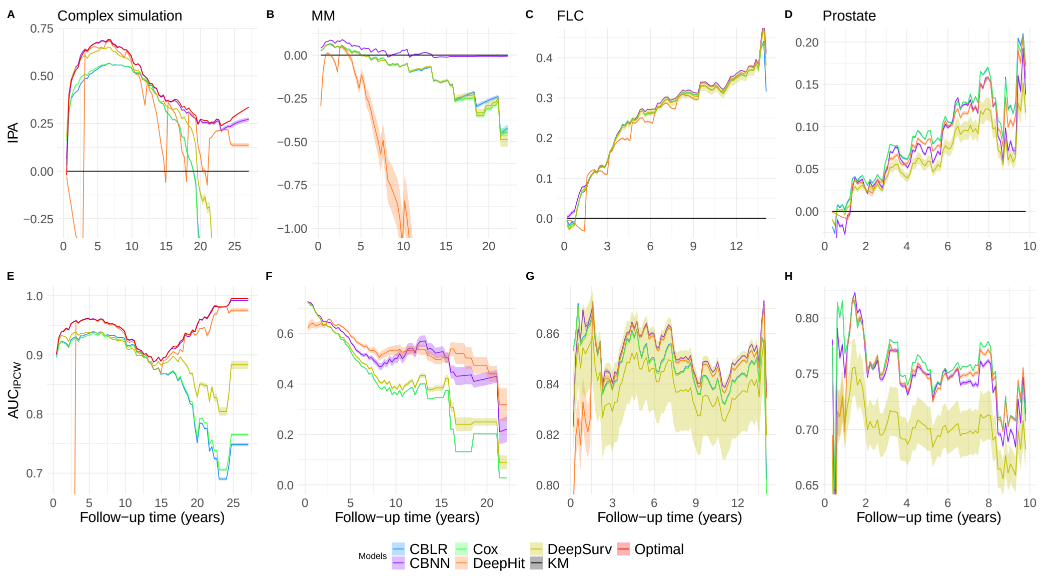

Figure 2: Performance of each model in the complex simulation (A, E), multiple myeloma (MM) case study (B, F), free light chain (FLC) case study (C, G) and prostate cancer (Prostate) case study (D, H). The first row shows the IPA for each model in each study over follow-up time. Negative values mean the model performs worse than the null model and positive values mean the model performs better. The second row shows the AUC_{IPCW} for each model in each study over follow-up time, where higher is better. Each model-specific metric in each study shows a 95% confidence interval over 100 iterations. Metrics are shown for six models: Case-Base with Logistic Regression (CBLR), Case-Base Neural Network (CBNN), Cox Proportional Hazard (Cox), DeepHit and DeepSurv. The Kaplan-Meier (KM) model serves as a baseline, predicting the average curve for all individuals. CBLR and Cox have near identical performance, resulting in curves that overlap. The Optimal model (a CBLR model with the exact interaction terms and baseline hazard specified) shows the best performance we can expect on the simulated data.

3. Simulation study

We simulate data to evaluate the performance of CBNN in comparison with existing regression (Cox, CBLR) and neural network (DeepHit, DeepSurv) methods. We specify a linear combination of each covariate as the linear predictor in the regression-based methods (Cox, CBLR), which contrasts with neural network approaches that allow for approximations of non-linear interactions. We simulate data based on a complex baseline hazard with time-varying interactions and 10% random censoring. We simulate three covariates for 5000 individuals:

$$z_1 \sim \text{Bernoulli}(0.5) \qquad z_2 \sim \begin{cases} N(0, 0.5) & \text{if } z_1 = 0 \\ N(1, 0.5) & \text{if } z_1 = 1 \end{cases} \qquad z_3 \sim N(1, 0.5).$$

In addition to the methods described above, we include the exact functional form of the covariates in a CBLR model (referred to as Optimal for simplicity) in the complex simulation. We obtain confidence intervals by conducting 100 bootstrap re-samples on the training data. We keep 15% of the data for testing before hyperparameter selection. We use 15% of the remaining data for validation and the rest is reserved for training. We predict risk functions for individuals in the test set, which are used to calculate our IPA and AUC_{IPCW}.

3.1. Complex simulation: flexible baseline hazard, time-varying interactions

We use this simulation to assess performance on data with a complex baseline hazard and a time-varying interaction. We design the model

$$\log h(t \mid X_i) = \sum_{i=1}^{5} (\gamma_i \cdot \psi_i) + \beta_1(z_1) + \beta_2(z_2) + \beta_3(z_3) + \tau_1(z_1 \cdot t) + \tau_2(z_2 \cdot z_3),$$

where $\gamma_1 = 3.9, \gamma_2 = 3, \gamma_3 = -0.43, \gamma_4 = 1.33, \gamma_5 = -0.86, \beta_1 = -5, \beta_2 = -1, \beta_3 = 1, \tau_1 = 0.001, \tau_2 = -1$ and ψ_i are basis splines. The γ coefficients are obtained from an intercept-only cubic splines model with three knots using the *flexsurvspline* function from the **flexsurv** package [23] on the German Breast Cancer Study Group dataset as it provides a complex baseline hazard from which we can simulate. The study comprised 686 women with breast cancer followed between 1984 and 1989 [3]. These γ, β and τ coefficients are used as our baseline hazard parameters and are fixed for the analysis. The β coefficients represent direct effects, τ_2 represents an interaction and τ_1 is a time-varying interaction.

3.2. Performance comparison in complex simulation

Figure 2 A, E and Table 2 shows the performance over time on a test set. The Optimal model acts as a reference for ideal performance on the simulated data. For discrimination, the Optimal model performs best, followed by CBNN, DeepHit, DeepSurv and the linear models (Figure 2 E). To obtain a more realistic performance assessment, we compared models in three case studies with a time-to-event outcome.

Table 2: Performance at certain percentages of follow-up time in the complex simulation (A), multiple myeloma (MM) case study (B), free light chain (FLC) case study (C) and prostate cancer (Prostate) case study (D). Each table shows performance for each method in each study at 25%, 50%, 75% and 100% of follow-up time. These tables are included to provide exact measures of performance. The models of interest are case-base with logistic regression (CBLR), Cox, Case-Base Neural Network (CBNN), DeepHit, DeepSurv, and Optimal (in the complex simulation). The best score at each percent of follow-up time is highlighted in bold. If the average performance is tied, then all tied values are highlighted.

A:Complex	IPA				AUC_{IPCW}			
Method	25%	50%	75%	100%	25%	50%	75%	100%
Cox	0.56 (0.56,0.57)	0.41.00 (0.4,0.41)	-0.53 (-0.54,-0.52)	-1.87 (-1.89,-1.84)	0.93 (0.93,0.94)	0.89 (0.88,0.89)	0.79 (0.79,0.79)	0.77 (0.76,0.77)
CBLR	0.56 (0.56,0.57)	0.4 (0.4,0.41)	-0.46 (-0.46,-0.45)	-1.5 (-1.52,-1.48)	0.94 (0.94,0.94)	0.89 (0.89,0.89)	0.78 (0.78,0.79)	0.75 (0.75,0.75)
DeepSurv	0.65 (0.65,0.65)	0.4 (0.4,0.4)	-0.16 (-0.18,-0.13)	-1.02 (-1.08,-0.95)	0.94 (0.94,0.94)	0.89 (0.89,0.89)	0.85 (0.85,0.85)	0.88 (0.88,0.89)
DeepHit	0.68 (0.68,0.68)	0.3 (0.3,0.31)	0.00 (-0.01,0.02)	0.14 (0.13,0.15)	**0.96 (0.96,0.96)**	0.88 (0.88,0.89)	0.94 (0.94,0.95)	0.98 (0.97,0.98)
CBNN	**0.69 (0.69,0.69)**	0.43 (0.43,0.43)	0.25 (0.25,0.26)	0.27 (0.26,0.28)	**0.96 (0.96,0.96)**	**0.9 (0.9,0.9)**	**0.96 (0.96,0.96)**	0.99 (0.99,0.99)
Optimal	**0.69 (0.69,0.69)**	**0.44 (0.43,0.44)**	**0.27 (0.27,0.27)**	**0.34 (0.33,0.34)**	**0.96 (0.96,0.96)**	**0.9 (0.9,0.9)**	**0.96 (0.96,0.96)**	**1.00 (1.00,1.00)**

B:MM	IPA				AUC_{IPCW}			
Method	25%	50%	75%	100%	25%	50%	75%	100%
Cox	0.00 (0.00,0.00)	-0.09 (-0.1,-0.09)	-0.26 (-0.27,-0.25)	-0.45 (-0.47,-0.42)	0.51.00 (0.51,0.51)	0.37 (0.37,0.37)	0.13 (0.13,0.13)	0.03 (0.03,0.03)
CBLR	0.00 (0.00,0.00)	-0.1.00 (-0.1,-0.09)	-0.25 (-0.26,-0.24)	-0.42 (-0.44,-0.4)	0.51.00 (0.51,0.51)	0.37 (0.37,0.37)	0.13 (0.13,0.13)	0.03 (0.03,0.03)
DeepSurv	0.00 (-0.01,0.00)	-0.09 (-0.09,-0.08)	-0.24 (-0.26,-0.22)	-0.49 (-0.53,-0.45)	0.52 (0.52,0.53)	0.39 (0.39,0.4)	0.24 (0.21,0.27)	0.09 (0.06,0.12)
DeepHit	-0.16 (-0.2,-0.12)	-1.21.00 (-1.42,-1)	-3.86 (-4.4,-3.31)	-3.66 (-4.17,-3.15)	**0.59 (0.58,0.59)**	**0.53 (0.51,0.55)**	**0.52 (0.48,0.56)**	**0.32 (0.25,0.38)**
CBNN	**0.04 (0.04,0.04)**	**0.00 (-0.01,0.00)**	**-0.01.00 (-0.01,-0.01)**	**-0.01.00 (-0.01,-0.01)**	0.55 (0.55,0.56)	0.51.00 (0.49,0.53)	0.43 (0.4,0.47)	0.22 (0.17,0.27)

C:FLC	IPA				AUC_{IPCW}			
Method	25%	50%	75%	100%	25%	50%	75%	100%
Cox	**0.19 (0.19,0.19)**	0.29 (0.29,0.29)	0.33 (0.33,0.33)	**0.44 (0.44,0.44)**	**0.85 (0.85,0.85)**	0.85 (0.85,0.86)	0.84 (0.84,0.84)	0.8 (0.79,0.8)
CBLR	**0.19 (0.19,0.19)**	**0.3 (0.29,0.3)**	0.33 (0.33,0.33)	0.32 (0.31,0.32)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	0.84 (0.84,0.84)	0.8 (0.8,0.8)
DeepSurv	**0.19 (0.18,0.2)**	0.29 (0.28,0.31)	0.33 (0.32,0.34)	0.38 (0.33,0.43)	0.84 (0.82,0.85)	0.85 (0.83,0.87)	0.83 (0.82,0.85)	0.82 (0.8,0.84)
DeepHit	0.18 (0.18,0.19)	0.29 (0.29,0.29)	0.33 (0.33,0.33)	0.41.00 (0.36,0.46)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	**0.85 (0.85,0.85)**	0.83 (0.83,0.84)
CBNN	**0.19 (0.19,0.2)**	**0.3 (0.3,0.3)**	**0.34 (0.34,0.34)**	0.42 (0.41,0.42)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	**0.85 (0.85,0.85)**	**0.84 (0.83,0.84)**

D:Prostate	IPA				AUC_{IPCW}			
Method	25%	50%	75%	100%	25%	50%	75%	100%
Cox	**0.04 (0.04,0.04)**	**0.09 (0.09,0.09)**	**0.14 (0.14,0.14)**	**0.17 (0.17,0.17)**	**0.77 (0.76,0.77)**	**0.75 (0.75,0.75)**	**0.76 (0.76,0.76)**	0.71.00 (0.71,0.71)
CBLR	**0.04 (0.04,0.04)**	**0.09 (0.09,0.09)**	**0.14 (0.13,0.14)**	**0.17 (0.17,0.17)**	**0.77 (0.76,0.77)**	**0.75 (0.75,0.75)**	**0.76 (0.76,0.76)**	0.71.00 (0.71,0.71)
DeepSurv	0.03 (0.03,0.04)	0.06 (0.05,0.07)	0.1.00 (0.09,0.11)	0.12 (0.1,0.13)	0.71.00 (0.69,0.73)	0.7 (0.68,0.72)	0.7 (0.68,0.72)	0.68 (0.66,0.7)
DeepHit	**0.04 (0.03,0.04)**	0.08 (0.08,0.09)	0.12 (0.12,0.13)	0.16 (0.15,0.16)	**0.77 (0.76,0.77)**	**0.75 (0.74,0.75)**	0.75 (0.75,0.75)	**0.72 (0.72,0.73)**
CBNN	**0.04 (0.04,0.04)**	0.08 (0.07,0.08)	0.12 (0.12,0.13)	0.14 (0.13,0.14)	0.76 (0.76,0.76)	**0.75 (0.75,0.75)**	0.74 (0.74,0.74)	0.71.00 (0.71,0.72)

4. Case studies

The simulation examines whether a complex baseline hazard and time-varying interactions affect method performance. The case study assesses performance in more realistic conditions, where we may not know if a flexible baseline hazard is beneficial, or if time-varying covariates exist. For each dataset, we perform a grid search with the same hyperparameter procedure as described in the simulation (three-fold cross-validated grid search). We then predict risk functions for everyone in the test set with the selected hyperparameters, which is used to calculate our metrics. We conduct 100-fold bootstrap re-samples on the training data to get confidence intervals.

4.1. Performance evaluation on multiple myeloma dataset

We expect cancer may have risk factors that vary with time [2]. As such, the first case study examines multiple myeloma (MM), using a cohort of 3882 patients seen at the Mayo Clinic from 1947 to 1996 until death [26]. Data are provided by the **survival** package [19]. We use two covariates, year of entry into the study and the time of MM diagnosis [26]. We see 71% incidence over 23 years [26].

Figure 2 B, F and Table 2 B demonstrate the performance over time on a test set. For discrimination, CBNN and DeepHit have a similar performance, followed by DeepSurv and finally the linear models performing worst (Figure 2 F). For both discrimination and calibration, CBNN performs substantially better than the linear models and DeepSurv, with DeepHit having the worst performance overall (Figure 2 B). Together, CBNN is the best calibrated model and one of the best at discrimination in the MM case study.

4.2. Performance evaluation on free light chain dataset

Serum free light chain (FLC) is a known diagnostic tool for assessing MM [27]. We are interested in whether there is a predictive benefit if the FLC markers varies with time. As such the second case study examines the relationship between serum FLC and mortality in a random sample of half the individuals in $\frac{2}{3}$ of the residents of Olmsted County over the age of 50 [28]. Data are provided by the **survival** package [19] for 7874 subjects tracked until death with 27% incidence over 14 years [28]. We use five covariates, total serum FLC (sum of kappa and lambda), age, sex, serum creatine and monoclonal gammopathy state [28].

Figure 2 C, G and Table 2 C demonstrate the performance over time on a test set. CBNN, DeepHit and the linear models perform best at discrimination, followed by DeepSurv (Figure 2 G). The rankings remain the same aside from DeepHit where the performance periodically drops to worse than DeepSurv (Figure 2 C). Together, CBNN outperforms the competing models in terms of both calibration and discrimination. However, CBNN is only slightly better than the linear ones.

4.3. Performance evaluation on prostate cancer dataset

As the CBNN model saw large predictive benefits on MM and small predictive benefits on FLC, we wish to examine longitudinal data with a more complex risk profile. The third case study (Prostate) examines prostate cancer survival on a publicly available simulation of the Surveillance, Epidemiology, and End Results (SEER) medicare study [29]. The Prostate dataset, which is contained in the **asaur** package [30], has a record of competing risks. As we are only interested in the single event scenario, we only keep individuals with prostate cancer death or censoring. This subset tracks 11054 individuals with three covariates: differentiation grade, age group and cancer state. There is a 7% incidence over 10 years [29].

Figure 2 C, G and Table 2 C demonstrate the performance over time on a test set. In the prostate case study, the linear models outperform the other models in both discrimination and calibration, followed by CBNN and DeepHit and finally DeepSurv (Figure 2 C, G). Aside from DeepSurv, the performance is similar across all models.

5. Discussion

CBNNs model survival outcomes by using neural networks on case-base sampled data. We incorporate follow-up time as a feature, providing a data-driven estimate of a flexible baseline hazard and time-varying interactions in our hazard function. The two competing neural network models we evaluated cannot model time-varying interactions [4] [9]. The CBNN model performs better due to its ability to model time-varying interactions and a complex baseline hazard.

The complex simulation requires a method that can learn both time-varying interactions and have a flexible baseline hazard. Based on our complex simulation results (Figure 2 A, E and Table 2 A), CBNN outperforms the competitors. This simulation shows how all models perform under ideal conditions with minimal noise in the data, while the three case studies assess their performance in realistic conditions. In the MM case study, flexibility in both interaction modeling and baseline hazard improves the performance of CBNN over the other models, suggesting that this flexibility aids calibration (Figure 2 B, F and Table 2 B). Upon examination of the FLC case study, CBNN demonstrates a small improvement to performance compared to the linear models and DeepHit for both IPA and AUC (Figure 2 C, G and Table 2 C). In the Prostate case study, the linear models outperform the neural network ones, while CBNN and DeepHit alternate their positions depending on the follow-up time of interest and DeepSurv maintains last place (Figure 2 C, G and Table 2 C). We attribute this to potential over-parameterization in the neural network models, as we did not test for fewer nodes in each hidden layer, even with dropout. Though the ranking places the linear models above the neural network ones, their overall performance falls within a small range of IPA and AUC values aside from DeepSurv.

Compared to CBNN, the neural network competitors also have limitations. DeepSurv is a proportional hazards model and does not estimate the baseline hazard [4]. DeepHit requires an alpha hyperparameter, assumes a single distribution for the baseline hazard and models the survival function directly [9]. The alternative neural network methods match on time, while CBNN models time directly. Another method caught our interest during our literature review. Deep Survival Machines (DSM), a parametric survival model using neural networks with a mixture of distributions that can fit a flexible baseline hazard [31]. With our hyperparameter options, DSM did not converge in the complex simulation which may have been due to issues in software or method specific limitations. Therefore, we did not include this method in our comparison.

While we apply a full grid search with three-fold cross-validation for all neural network models, there are an infinite number of untested hyperparameters we did not test. A completely exhaustive search is computationally infeasible, therefore it is reasonable to expect that there exists a set of hyperparameters that is better suited for each model in each study. However, we test a reasonably large range while accounting for potential over-fitting by including dropout [12]. We provide the same options to all models aside from DeepHit, which has a method specific hyperparameter α [9]. With these limitations in mind, we summarize the differences in performance across all tested methods.

If prediction is our goal, we suggest CBNN as the best model in the Complex simulation, MM case study and FLC case study. The linear models were competitive in the FLC case study and performed best in the Prostate case study. Though neural network interpretability is steadily improving [32], there is still a trade-off compared to regression models, especially when the predictive performance gain is minimal, like in the FLC case study. We suggest that reference models should be included when assessing neural network models. Both a null model (KM curve) and a linear model (either Cox or a flexible baseline hazard model like CBLR) provide insight as to whether the neural network model is learning anything useful beyond linear predictors that were not accounted for.

6. Conclusions

Our study was motivated by the lack of easily implemented survival models based on neural networks that can approximate time-varying interactions. This lead us to apply the case-base sampling technique, which has previously been used with logistic regression [1], to neural network models for a data-driven approach to time-varying interaction modeling. In our paper, We aim to compare CBNNs with existing methods. We assess the discrimination and calibration of each method in a simulation with time-varying interactions and a complex baseline hazard, and three case studies. CBNNs outperform all competitors in the complex simulation and two case studies while maintaining competitive performance in a final case study. Once we perform case-base sampling and adjust for the sampling bias, we can use a sigmoid activation function to predict our hazard function. Our approach provides an alternative to incorporating censored individuals, treating survival outcomes as binary ones. Forgoing the requirement of custom loss functions, CBNNs only require the use of standard components in machine learning libraries (specifically, the add layer to adjust for sampling bias and the sigmoid activation function) after case-base sampling. Due to the simplicity in its implementation and by extension user experience, CBNNs are both a user-friendly approach to data-driven, single event survival analysis and is easily extendable to any feed-forward neural network framework.

Data and code availability statement

The MM and FLC datasets are available in the **survival** package in R [19]. The Prostate dataset is available as part of the **asaur** package in R [30]. The code for this manuscript and its analyses can be found at `https://github.`

`com/Jesse-Islam/cbnnManuscript`. The software package making CBNN easier to implement can be found at `https://github.com/Jesse-Islam/cbnn`.

Acknowledgements

We would like to thank Dr. James Meigs, the project leader of UM1DK078616 and R01HL151855, for his support and helpful discussions. We would also like to thank Dr. James Hanley for his support and discussions while extending the case-base methodology.

Author contributions

J.I. conceived the proof of concept, developed the theoretical formalism, developed the neural network code base, wrote the majority of the manuscript and performed the simulations and analyses. M.T. wrote the majority of the case-base sampling section 2.1. M.T. and S.B. provided key insights into the core sampling technique (case-base). R.S. provided multiple edits to the manuscript structure and key insights. All authors discussed the results and contributed to the final manuscript.

Financial disclosure

This work was supported by subcontracts from UM1DK078616 and R01HL151855 to R.S. The work was also supported as part of the Congressionally Directed Medical Research Programs (CDMRP) award W81XWH-17-1-0347.

Conflict of interest

The authors declare no potential conflict of interests.

References

[1] J. A. Hanley, O. S. Miettinen, Fitting smooth-in-time prognostic risk functions via logistic regression, The International Journal of Biostatistics 5 (1) (2009).

[2] D. Coradini, M. G. Daidone, P. Boracchi, E. Biganzoli, S. Oriana, G. Bresciani, C. Pellizzaro, G. Tomasic, G. Di Fronzo, E. Marubini, Time-dependent relevance of steroid receptors in breast cancer, Journal of clinical oncology 18 (14) (2000) 2702–2709.

[3] P. Royston, M. K. Parmar, Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects, Statistics in medicine 21 (15) (2002) 2175–2197.

[4] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, Y. Kluger, Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network, BMC Medical Research Methodology 18 (1) (2018) 24.

[5] H. Yu, T. Huang, B. Feng, J. Lyu, Deep-learning model for predicting the survival of rectal adenocarcinoma patients based on a surveillance, epidemiology, and end results analysis, BMC cancer 22 (1) (2022) 1–14.

[6] N. Bice, N. Kirby, T. Bahr, K. Rasmussen, D. Saenz, T. Wagner, N. Papanikolaou, M. Fakhreddine, Deep learning-based survival analysis for brain metastasis patients with the national cancer database, Journal of applied clinical medical physics 21 (9) (2020) 187–192.

[7] D. W. Kim, S. Lee, S. Kwon, W. Nam, I.-H. Cha, H. J. Kim, Deep learning-based survival prediction of oral cancer patients, Scientific reports 9 (1) (2019) 1–10.

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

[9] C. Lee, W. R. Zame, J. Yoon, M. v. d. Schaar, Deephit: A deep learning approach to survival analysis with competing risks., AAAI Conference on Artificial Intelligence (2018) 2314–2321Software from pycox version 0.2.2.

[10] N. Mantel, Synthetic retrospective studies and related topics, Biometrics (1973) 479–486.

[11] O. Saarela, J. A. Hanley, Case-base methods for studying vaccination safety, Biometrics 71 (1) (2015) 42–52.

[12] A. Gulli, S. Pal, Deep learning with Keras, Packt Publishing Ltd, 2017.

[13] D. Hughes-Hallett, A. M. Gleason, W. G. McCallum, Calculus: Single and multivariable, John Wiley & Sons, 2020.

[14] E. Graf, C. Schmoor, W. Sauerbrei, M. Schumacher, Assessment and comparison of prognostic classification schemes for survival data, Statistics in medicine 18 (17-18) (1999) 2529–2545.

[15] M. W. Kattan, T. A. Gerds, The index of prediction accuracy: an intuitive measure useful for evaluating risk prediction models, Diagnostic and prognostic research 2 (1) (2018) 1–7.

[16] P. Blanche, C. Proust-Lima, L. Loubere, C. Berr, J.-F. Dartigues, H. Jacqmin-Gadda, Quantifying and comparing dynamic predictive accuracy of joint models for longitudinal marker and time-to-event in presence of censoring and competing risks, Biometrics 71 (1) (2015) 102–113.

[17] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, version 4.1.2 (2021).
URL `https://www.R-project.org/`

[18] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009, version 3.8.12.

[19] Terry M. Therneau, Patricia M. Grambsch, Modeling Survival Data: Extending the Cox Model, Springer, New York, 2000, r package version 3.2-11.

[20] S. R. Bhatnagar, M. Turgeon, J. Islam, J. A. Hanley, O. Saarela, casebase: An alternative framework for survival analysis and comparison of event rates, arXiv preprint arXiv:2009.10264 (2020).

[21] J. Allaire, F. Chollet, keras: R Interface to 'Keras', r package version 2.7.0 (2021).
URL `https://CRAN.R-project.org/package=keras`

[22] S. L. Brilleman, R. Wolfe, M. Moreno-Betancur, M. J. Crowther, Simulating survival data using the simsurv R package, Journal of Statistical Software 97 (3) (2020) 1–27. `doi:10.18637/jss.v097.i03`.

[23] C. Jackson, flexsurv: A platform for parametric survival modeling in R, Journal of Statistical Software 70 (8) (2016) 1–33, r package version 2.0. `doi:10.18637/jss.v070.i08`.

[24] T. A. Gerds, M. W. Kattan, Medical Risk Prediction Models: With Ties to Machine Learning (1st ed.), Chapman and Hall/CRC, 2021, r package version 2021.10.10.
URL `https://doi.org/10.1201/9781138384484`

[25] K. Ushey, J. Allaire, Y. Tang, reticulate: Interface to 'Python', r package version 1.22 (2021).
URL `https://CRAN.R-project.org/package=reticulate`

[26] R. Kyle, Long term survival in multiple myeloma, New Eng J Medicine (1997).

[27] A. Dispenzieri, R. Kyle, G. Merlini, J. Miguel, H. Ludwig, R. Hájek, A. Palumbo, S. Jagannath, J. Bladé, S. Lonial, et al., International myeloma working group guidelines for serum-free light chain analysis in multiple myeloma and related disorders, Leukemia 23 (2) (2009) 215–224.

[28] A. Dispenzieri, J. A. Katzmann, R. A. Kyle, D. R. Larson, T. M. Therneau, C. L. Colby, R. J. Clark, G. P. Mead, S. Kumar, L. J. Melton III, et al., Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population, in: Mayo Clinic Proceedings, Vol. 87, Elsevier, 2012, pp. 517–523.

[29] G. L. Lu-Yao, P. C. Albertsen, D. F. Moore, W. Shih, Y. Lin, R. S. DiPaola, M. J. Barry, A. Zietman, M. O'Leary, E. Walker-Corkery, et al., Outcomes of localized prostate cancer following conservative management, Jama 302 (11) (2009) 1202–1209.

[30] D. F. Moore, asaur: Data Sets for "Applied Survival Analysis Using R"", r package version 0.50 (2016).
URL `https://CRAN.R-project.org/package=asaur`

[31] C. Nagpal, X. R. Li, A. Dubrawski, Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks, IEEE Journal of Biomedical and Health InformaticsSoftware downloaded March 10, 2021 (2021).

[32] Y. Zhang, P. Tiňo, A. Leonardis, K. Tang, A survey on neural network interpretability, IEEE Transactions on Emerging Topics in Computational Intelligence 5 (5) (2021) 726–742.