Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

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

Jesse Islam[1] | Maxime Turgeon[2] | Robert Sladek[3] | Sahir Bhatnagar[4]

[1]Department of Quantitative life sciences, McGill University, Quebec, Canada
[2]Department of Statistics, University of Manitoba, Manitoba, Canada
[3]Department of Human Genetics, McGill University, Quebec, Canada
[4]Department of Biostatistics, McGill University, Quebec, Canada

Correspondence
Jesse Islam's Email:
jesse.islam@mail.mcgill.ca

Present Address
This is sample for present address text this is sample for present address text

Summary

Neural network-based survival methods can model data-driven covariate interactions. While these methods have led to better predictive performance than regression-based approaches, they cannot model both time-varying interactions and complex baseline hazards. To address this, we propose Case-Base Neural Networks (CBNN) as a new approach that combines the case-base sampling framework with flexible architectures. Our method naturally accounts for censoring and does not require method specific hyperparameters. Using a novel sampling scheme and data augmentation, we incorporate time directly into a feed-forward neural network. CBNN predicts the probability of an event occurring at a given moment and estimates the hazard function. We compare the performance of CBNN to survival methods based on regression and neural networks in a simulation and three real data applications. We report two time-dependent metrics for each model. In the simulations and real data applications, CBNN provides a more consistent predictive performance across time and outperforms the competing neural network approaches. For a complex simulation, which highlights the ability of CBNN to model both a complex baseline hazard and time-varying interactions, CBNN outperforms all competitors. The first two real data application shows CBNN outperforming all neural network competitors, while a second real data application shows competitive performance. We 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 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

Smooth-in-time accelerated failure time (AFT) models can estimate absolute risks by modeling the hazard directly through a user-specified baseline hazard distribution (Kleinbaum & Klein 2012). Cox proportional hazards models are used more often than AFT models, causing analyses to be based on hazard ratios and relative risks rather than on survival curves and absolute risks (Hanley & Miettinen 2009). The identification of an appropriate distribution for the baseline hazard in an AFT model may be difficult for common diseases that have many interacting risk factors, or a Cox model where the disease pathogenesis may change with age, making it difficult to maintain the proportional hazards assumption. For example, previous studies of breast cancer incidence have discovered time-varying interactions with covariates of interest, such as tumor size (Coradini et al. 2000). One approach to provide flexibility in the baseline hazard involves using the basis of splines on time in our model (Royston & Parmar 2002).

However, regression-based models are limited in that they require prior knowledge about potential time-varying interactions and their quantitative effects.

Neural networks provide a data-driven approach to approximating interaction terms. For example, DeepSurv is a neural network-based proportional hazards model that implements the Cox partial log-likelihood as a custom loss function (Katzman et al. 2018), resulting in a stepwise absolute risk curve that cannot accommodate time-varying interactions. Compared to Cox regression, DeepSurv shows better performance on a real dataset (Katzman et al. 2018). To handle non-proportional hazards, a modification to the loss function was proposed (Faraggi & Simon 1995).

As an alternative method that assumes a baseline hazard distribution, DeepHit specifies each survival time of interest in the model and directly estimates survival curves, rather than deriving a hazard function . It assumes an inverse Gaussian distribution as the baseline hazard and it outperformed DeepSurv (Lee, Zame, Yoon, & Schaar 2018).

Providing further flexibility, Deep Survival Machines (DSM) is a parametric survival model using neural networks with a mixture of distributions as the baseline hazard (Nagpal, Li, & Dubrawski 2021). DSM outperformed DeepSurv and DeepHit (Nagpal et al. 2021). However, like DeepHit and DeepSurv, DSM cannot model time-varying interactions.

We note that these alternative neural network approaches all require custom loss functions (Katzman et al. 2018) (Lee et al. 2018) (Nagpal et al. 2021). DeepHit introduces a hyperparameter weighing its two loss functions (negative log-likelihood and ranking losses) while DSM requires a two-phase learning process and user implementations for distributions beyond Log-Normal or Weibull (Lee et al. 2018) (Nagpal et al. 2021). Regression-based approaches require prior specification of all interaction terms, which makes it challenging to model covariate effects that change over time. The current neural network models provide flexibility at the cost of opacity, while regression models provide clarity at the cost of flexibility.

In this article, we propose Case-Base Neural Networks (CBNN) as a method that models time-varying interactions and a flexible baseline hazard using commonly available neural network components. Our approach to modeling the full hazard uses case-base sampling (Hanley & Miettinen 2009). This sampling technique allows probabilistic models to predict survival outcomes. As part of the case-base framework, we use transformations of time as a feature (covariate) to specify different baseline hazards. For example, by including splines of time as covariates, we can approximate the Royston-Parmar flexible baseline hazard model (Royston & Parmar 2002), however, this still requires explicit use of time-varying interactions. CBNN can model both without extra tuning parameters.

In Section 2, we describe how case-base sampling and neural networks are combined both conceptually and algebraically, along with our hyperparameter choices and software implementation. Section 4, describes our metrics and compares the performance of CBNN, DeepSurv, DeepHit, DSM, Cox regression and case-base using logistic regression (CBLR) on simulated data. Section 5 describes the real-data analysis, while Section 6 explores the implications of our results and contextualizes them within neural network survival analysis in a single event setting.

2 | CASE-BASE NEURAL NETWORK METHODOLOGY, METRICS AND SOFTWARE

In this section, we define case-base sampling, which converts the total survival time into discrete person-specific moments (person-moments). Then, we detail how neural networks can be used within this framework, explicitly incorporating time as a feature while adjusting for the sampling bias. Finally, we report on the software versions used. An R package is available for use 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

Case-base sampling is an alternative framework for survival analysis (Hanley & Miettinen 2009). In case-base sampling, we sample from the continuous survival time of each person in our dataset to create a *base series* of *person-moments*. 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 potential 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. 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

Figure 1 Steps involved in CBNN from case-base sampling to the model framework we use for training. The first step is case-base sampling, completed before training begins. Next, we pass this sampled data through a feed-forward neural network. We add the offset and pass that through a sigmoid activation function, whose output is a probability. Once the neural network model completes its training, we can convert the probability output to a hazard, using it for our survival outcomes of interest.

equality [1]:

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

The odds of a person-moment being a part of 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 adjusting for the bias introduced when sampling a fraction of the study base B by adding the offset term $\log\left(\frac{b}{B}\right)$ as in (2). Next, we propose using neural networks to model the odds.

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, to which an offset term is added, then passed through a sigmoid activation function (Figure 1). We use the sigmoid function as its inverse is the odds, which we can use to calculate the hazard. The general form for the neural network using CBNN is:

$$P\left(Y = 1|X, T\right) = \text{sigmoid}\left(f_\theta(X, T) + \log\left(\frac{B}{b}\right)\right), \tag{3}$$

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 bias term 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, where user-defined hyperparameters such as regularization, number of layers and nodes control the flexibility of the hazard function. We provide a detailed description of the choices we made in the next sub-section.

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

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

We use binary cross-entropy as our loss function (Gulli & Pal 2017):

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

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 (e.g. Adam, RMSPropagation, stochastic gradient descent) is used to optimize the parameters in the model (Gulli & Pal 2017). For our analysis, we use Adam as implemented in Keras (Gulli & Pal 2017). Note that the size of the *case*

[1] We are abusing notation here, conflating hazards with probabilities. For a rigorous treatment, see Saarela & Hanley (2015) section 3 Saarela and Hanley (2015) .

series is fixed as the number of events, but we can make the *base series* as large as we want. A ratio of 100:1 *base series* to *case series* is sufficient (Hanley & Miettinen 2009). We pass our feed-forward neural network through a sigmoid activation function (Figure 1). Finally, we can convert this model output to a hazard. When using our model for predictions, we manually set the offset term to 0 in the new data, as we account for the 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, viz.

$$F\left(t \mid X\right) = 1 - \exp\left(-\int_0^t h(u|X)\,\mathrm{d}u\right). \tag{4}$$

We use a finite Riemann sum (Hughes-Hallett, Gleason, & McCallum 2020) to approximate the integral in (4).

2.3 | Hyperparameter selection

Neural networks are flexible when defining the architecture and optimization parameters. These hyperparameter choices can affect the estimated parameters and are not necessarily the same for each model. First we fix a test set with 15% of the data, then we use three-fold cross-validation on the remaining data (85%:15% training:validation) for a range of hyperparameters (TABLE REF). We track integrated brier score on the validation set for each hyperparameter combination, choosing the combination with the lowest score for each method. The chosen hyperparameters are in (TABLE REF).

2.4 | Software implementation

R (R Core Team 2021) and python (Van Rossum & Drake 2009) are used to evaluate methods from both languages. We fit the Cox model using the **survival** package (Terry M. Therneau & Patricia M. Grambsch 2000), the CBLR model using the **casebase** package (Bhatnagar, Turgeon, Islam, Hanley, & Saarela 2020), the DeepSurv model using the **survivalmodels** package (Sonabend 2021), the DeepHit model using **pycox** (Lee et al. 2018) and the DSM model using **DeepSurvivalMachines** (Nagpal et al. 2021). We made the components of CBNN using the **casebase** package for the sampling step and the **keras** (Allaire & Chollet 2021) package for our neural network architecture. The **simsurv** package (Brilleman, Wolfe, Moreno-Betancur, & Crowther 2020) is used for our simulation studies, while **flexsurv** (Jackson 2016) is used to fit a flexible baseline hazard using splines for our complex simulation. We use the implementation of C_{IPCW} from the python package **sksurv** (Pölsterl 2020). The **riskRegression** package (Gerds & Kattan 2021) is used to get the Index of Prediction Accuracy (IPA metric). Both metrics are described in detail in the following section. We modify the **riskRegression** package to be used with any user supplied risk function F. To ensure that both R and Python-based models are running in unison on the same data through our simulations and bootstrap, we use the **reticulate** package (Ushey, Allaire, & Tang 2021).

3 | PERFORMANCE METRICS

To choose our method-specific hyperparameters, we use the Integrated Brier Score (IBS) (Graf, Schmoor, Sauerbrei, & Schumacher 1999). We use two metrics to assess the performance of the different methods of interest on testing data: 1) (IPA) (Kattan & Gerds 2018) and 2) inverse probability censoring weights-adjusted concordance index (C_{IPCW}) (Uno, Cai, Pencina, D'Agostino, & Wei 2011), which we define below. IBS provides a summarized assessment of performance for each model. (C_{IPCW}) provides information about performance up to the survival time of interest. while IPA provide transparency as to when in follow-up time each model may perform better than the others.

3.0.1 | Index of prediction accuracy (IPA)

The IPA is a function of the Brier score ($BS(t)$) (Graf et al. 1999), which is defined as

$$BS(t) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\left(1-\widehat{F}(t \mid X_i)\right)^2 \cdot I(T_i \leq t, \delta_i = 1)}{\widehat{G}(T_i)} + \frac{\widehat{F}(t \mid X_i)^2 \cdot I(T_i > t)}{\widehat{G}(t)}\right), \tag{5}$$

where $\delta_i = 1$ shows individuals who have experienced the event, N represents the number of samples in our dataset over which we calculate $BS(t)$, $\widehat{G}(t) = P[c > t]$ is a non-parametric estimate of the censoring distribution, c is censoring time and T_i is an individual's survival or censoring time. The Brier score provides a score that accounts for the information loss because of censoring. There are three categories of individuals that may appear within the dataset once we fix our t of interest. 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 are

Figure 2 Summarizes the simple simulation (A, B), complex simulation (C, D), SUPPORT case study (E, F) and METABRIC case study (G, H) results. The first row shows the IPA score for each model in each study over follow-up time. Negative values mean our model performs worse than the null model and positive values mean the model performs better. The second row demonstrates the C_{IPCW} score for each model in each study over follow-up time. A score of 1 is the maximum performance for either metric. Each model-specific metric in each study shows a 95-percent confidence interval over 100 iterations. The models of interest are case-base with logistic regression (CBLR), Case-Base Neural Networks (CBNN), Cox, DeepHit, DeepSurv, Deep Survival Machines (DSM), Optimal (a CBLR model with the exact interaction terms and baseline hazard specified) and Kaplan-Meier (to serve as a baseline, predicting the average for all individuals).

the third category of people. The inverse probability censoring weights (IPCW) adjustment ($G(\cdot)$) is to account for these category three individuals whose information is missing. The IPA score as a function of time is given by

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

where $BS_{model}(t)$ represents the Brier score over time t for the model of interest and $BS_{null}(t)$ represents the Brier score if we use an unadjusted Kaplan-Meier (KM) curve as the prediction for all observations (Kattan & Gerds 2018). Note that IPA 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. These scores show how performance changes over follow-up time.

A potential artifact of IPA is that the score is unstable at earlier and later survival times. This is because of a near equivalent Brier score among each model and the null model. At small values, a difference of 0.1 creates a more significant fold change than at larger values. As the Brier score is potentially small at the start and at the end of their respective curves, The IPA score may be unstable at the same locations.

3.0.2 | Integrated Brier Score (IBS)

The IBS is a function of the Brier score ($BS(t)$) (Graf et al. 1999). as well, which is defined as

$$IBS = \int_{t}^{t_{max}} BS(t)dw(t) \tag{6}$$

where $w(t) = \frac{t}{t_{max}}$. As this is a score for a range of follow-up-time, it is a useful metric to track during cross-validation. The performance at a specific time is hidden, and is why we opt for the IPA in assessing the final models on a test set.

3.0.3 | Inverse probability censoring weights-adjusted concordance index

The C_{IPCW} is a non-proper, rank-based metric that does not depend on the censoring times in the test data (Uno et al. 2011). The C_{IPCW} is given by

$$C_{IPCW}(t) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_i\left\{\widehat{G}(T_i)\right\}^{-2} I(T_i < T_j, T_i < t) I\left(\widehat{F}(t|X_i) > \widehat{F}(t|X_j)\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_i\left\{\widehat{G}(T_i)\right\}^{-2} I(T_i < T_j, T_i < t)}. \tag{7}$$

where the follow-up period of interest is (0,t), $I(\cdot)$ is an indicator function, $\widehat{F}(X_i, t)$ is the risk function estimated for everyone in the study at time t and C_{IPCW} can compare the performance of different models, where a higher score is better. Note that the C_{IPCW} may produce misleading performance, as it ranks based on survival times, not event status (Blanche, Kattan, & Gerds 2019). This metric is considered an unbiased population concordance measure because of the IPCW adjustment (Uno et al. 2011).

4 | SIMULATION STUDY

In this section, we simulate data to evaluate the performance of CBNN and compare our approach with existing regression-based (Cox, CBLR) and neural network-based (DeepHit, DeepSurv) methods. We specify a linear combination of each covariate as the linear predictor in regression-based approaches (Cox, CBLR), which contrasts with neural network approaches that allow for non-linear interactions. We simulate data under a complex baseline hazard with time-varying interactions, each with 10% random censoring. For both settings, we simulate three covariates:

Table 1 Four tables representing performance at certain follow-up times for the simple simulation, complex simulation, SUPPORT and METABRIC. Each table shows performance for each method in each study at 25%, 50%. 75% and 100% of follow-up time. The bold elements show the best model for each study, at each follow-up time of interest. These tables are included to provide exact measures at certain intervals. The models of interest are: Cox, case-base with logistic regression (CBLR), DeepSurv, DeepHit, Case-Base Neural Network (CBNN) and Optimal

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

Besides the methods mentioned above, we include the Optimal model in our comparisons using CBLR. That is, we include the exact functional form of the covariates in a CBLR model (referred to as Optimal for simplicity). We calculate t-based 95% confidence intervals using 100 replications of the simulated data. For all analyses, 15% of the data is kept for testing, while 15% of the remaining data is kept for validation. The remaining data is kept for training. We predict risk functions F using (4) for individuals in the test set, which are used to calculate our C_{IPCW} and IPA scores. We conduct 100 bootstrap re-samples on the training data to obtain confidence intervals.

4.1 | Complex simulation: flexible baseline hazard, time-varying interactions

This simulation demonstrates performance with the presence of a complex baseline hazard and a time-varying interaction. Inspired by a cancer treatment that initially reduces risk of death, but slowly returns as time progresses. Originally used to show the spline-based hazard model proposed by Royston and Parmar (Royston & Parmar 2002), the breast cancer dataset provides a complex hazard from which we simulate, available in the **flexsurv** R package (Jackson 2016). To increase the complexity of our data-generating mechanism for this simulation, we design the model as follows:

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

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 are basis splines. The $gamma$ coefficients are obtained from an intercept-only cubic splines model with three knots using the *flexsurvspline* function from the **flexsurv** package (Jackson 2016). Note that we fix these values for the analysis. The β coefficients represent direct effects, τ_2 and τ_3 represent interactions and τ_1 is a time-varying interaction.

We use a population time plot to visualize incidence density across $z1$ (FIGURE REF). z1 acts to mimic a treatment and control group. The control group $(z1 = 0)$ is not impacted by the time-varying interaction. The treatment group $(z1 = 1)$ initially acts to protect the group, while the risk slowly builds with time at the rate of the interaction term (τ_1). This creates a separation between the median time of death between these two groups due to the time-varying interaction. We expect all competing models (aside from the Optimal model and CBNN) to drop in performance for the second half of follow-up time.

4.1.1 | Performance comparison in complex simulation

Figure 2 A, E and Table 1 A demonstrates the performance over time on a test set. While examining the IPA score, the optimal regression model and CBNN perform near identically, leading in performance followed by DeepHit, the linear regression models and DeepSurv. We expected the Optimal model to perform best in both metrics. However, this was not the case for C_{IPCW} and may be due to an artifact of concordance-based metrics, where a misspecified model may perform better than a correctly specified one (Blanche et al. 2019). We attribute the performance of CBNN to its flexibility in modeling time-varying interactions and baseline hazard, flexibility the other neural network models do not have. We keep this misspecification issue in mind while interpreting the C_{IPCW} results in the case studies.

5 | CASESTUDIES

Our complex simulation demonstrates the superior performance of CBNN in ideal conditions with clean data. To obtain a more realistic performance assessment, we compared models using three real datasets with a time-to-event outcome. The first case study examines multiple myeloma (MM) (Kyle 1997). The second examines the relationship between serum free light chain (FLC) and mortality (Dispenzieri et al. 2012). The third examines

prostate cancer survival on a publicly available simulation of the Surveillance, Epidemiology, and End Results (SEER)-medicare prostate cancer data(Lu-Yao et al. 2009). As we do not know the true model for the real data, we exclude the Optimal model. After grid search is performed with the same hyperparameter options, We split the data the same way we did for the simulation (15% of the data is kept for testing, while 15% of the remaining data is kept for validation and the rest for testing). We predict risk functions for everyone in the test set, which is used to calculate our metrics. We conduct 100-fold bootstrap re-samples on the training data to obtain confidence intervals.

5.1 | Performance evaluation on multiple myeloma dataset

The (MM) dataset tracks 3882 patients seen at the Mayo Clinic from 1947-1996 until death (Kyle 1997). We use 2 covariates, year of entry into the study and the time of MM diagnosis(Kyle 1997). We see 71% incidence over 23 years(Kyle 1997).

Figure 2 B, F and Table 1 B demonstrates the performance over time on a test set. With the IPA score, CBNN outperforms The competing models, followed by the linear models, DeepSurv and DeepHit. We note that after 50% of survival time, all models aside from CBNN perform substantially worse than a null model. We note with C_{IPCW}, the linear models and DeepSurv perform equivalently, followed by CBNN and DeepHit.

5.2 | Performance evaluation on free light chain dataset

The FLC dataset is a random sample of half the individuals in $\frac{2}{3}$ of the residents of Olmsted County over the age of 50(Dispenzieri et al. 2012). We have access to 7874 subjects tracked until death(Dispenzieri et al. 2012). We use 5 covariates, total serum FLC (kappa+lambda), age, sex, serum creatine and monoclonal gammapothy state(Dispenzieri et al. 2012). we see 27% incidence over 14 years(Dispenzieri et al. 2012).

Figure 2 C, G and Table 1 C demonstrates the performance over time on a test set. With the IPA score, CBNN outperforms The competing models, followed by the linear models and lastly DeepSurv and DeepHit for the first 60% of follow-up time. After 60% of follow-up time, DeepHit has similar performance to CBNN. For the C_{IPCW} score, CBNN outperforms all models, followed by the linear models and finally DeepSurv. Note that for earlier ranges of survival time, DeepHit is the worst performing model. After 25% of follow-up time, DeepHit is in second place. In the FLC dataset, CBNN has a consistent top ranking for both IPA and C_{IPCW}.

5.3 | Performance evaluation on prostate cancer dataset

The prostate cancer dataset has a record of competing risks(Lu-Yao et al. 2009). As we are only interested in the single event scenario, we only keep individuals with prostate cancer death or censoring(Lu-Yao et al. 2009). This subset tracks 11054 individuals with three covariates, differentiation grade, age group and cancer state(Lu-Yao et al. 2009). There is a 7% incidence over 10 years.

Figure 2 C, G and Table 1 C shows the performance over time on a test set. With the IPA score, the linear models outperform the neural network ones, followed by CBNN and DeepHit, and finally DeepSurv. For the C_{IPCW} score, the linear models, CBNN, DeepHit perform near identically, side from DeepHit with an initial drop in performance. DeepSurv performed substantially worse than the other models.

6 | DISCUSSION

CBNN models 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 by design (Katzman et al. 2018) (Lee et al. 2018) With our hyperparameters options and model design, DSM did not converge in the complex simulation. Due to this issue limitation, we did not include this method. Compared to CBNN, all three neural network models also have limitations. DeepSurv is a proportional hazards model and does not estimate the baseline hazard (Katzman et al. 2018). DeepHit requires an alpha hyperparameter, is restricted to a single distribution for the baseline hazard and models the survival function directly (Lee et al. 2018). Though not included, DSM can model a flexible baseline hazard, however is not meant for time-varying interactions(Nagpal et al. 2021). The alternative neural network methods match on time, while CBNN models time directly.

To assess performance among these models, we use both IPA and C_{IPCW} metrics. Concordance-based measures are commonly used in survival analysis to compare models and we opt to keep them in our analyses. However, C_{IPCW} is a non-proper metric and may cause misspecified models to appear better than they should (Blanche et al. 2019). Therefore, we contextualize our C_{IPCW} results in relation to IPA, a proper scoring rule. The model rankings between C_{IPCW} and IPA differed for all but FLC. If we consider Figure 2 A, E and Table 1 A in conjuction with FIGURE REF POPTIME, We note a universal drop in C_{IPCW} performance for all models. We hide the confidence intervals in Figure 2 E due to visibility but list

them in Table 1 A . aside from CBNN, all models have a wide confidence band at 75% of follow-up time. We expect the optimal model to perform best, but that's not the case looking at C_{IPCW}. As such, we

With this interpretation of C_{IPCW} in mind, we assess a simulations with minimal noise and three case studies.

The complex simulation requires method that can learn both time-varying interactions and have a flexible baseline hazard. Here, CBNN demonstrates a distinct advantage over all other methods. Based on our complex simulation results (Figure 2 C, D and Table 1 B), CBNN outperforms the competitors when time-varying interactions and a complex baseline hazard are present (after 50% of survival time, Figure REF POPTIME). This simulation shows how CBNN can perform under ideal conditions, while the three case studies serve to assess its performance in realistic conditions.

In the MM and FLC case studies, flexibility in both interaction modeling and baseline hazard improved CBNN's performance over the other models, suggesting that this flexibility aids prediction in both case studies. In the Prostate case study, the linear models outperform the neural network ones. CBNN and DeepHit alternate their positions depending on the survival time and DeepSurv maintains last place. We attribute this to potential overparameterization in the neural network models as we did not test for a small number of nodes in each hidden layer, even with dropout. Though the ranking places the linear models above the neural network ones, the overall performance is comparable aside from DeepSurv, falling within a small range of IPA scores.

7 | CONCLUSIONS

CBNN outperforms all competitors in the complex simulation and two case studies, while maintaining competitive performance in a final case study, demonstrating its value in survival settings that may involve time-varying interactions and a complex baseline hazard. 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 the incorporation of censored individuals, treating survival outcomes as binary ones. Forgoing the requirement of custom loss functions, CBNN only requires the use of standard components in machine learning libraries (specifically, the add layer to adjust for sampling bias and the sigmoid activation function). Due to the simplicity in its implementation and by extension user experience, CBNN is both a user-friendly approach to data-driven survival analysis and is easily extendable to any feed-forward neural network framework.

Data and code availability statement

The pre-processed data for the SUPPORT case study can be found at `https://github.com/jaredleekatzman/DeepSurv/tree/master/experiments/data/support`. The pre-processed data for the METABRIC case study can be found at `https://github.com/jaredleekatzman/DeepSurv/tree/master/experiments/data/metabric`. The data is accessed using `https://github.com/havakv/pycox`. 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 use can be found at `https://github.com/Jesse-Islam/cbnn`.

Acknowledgements

We would like to thank Dr. James Meigs, the project leader of these awards, for his support and helpful discussions. The work was also supported as part of the Congressionally Directed Medical Research Programs (CDMRP) award W81XWH-17-1-0347. 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. and S.B. provided key insights into the core sampling technique (casebase). All authors discussed the results and contributed to the final manuscript. M.T. contributed heavily to the casebase sampling section 2.1.

Financial disclosure

This work was supported by subcontracts from UM1DK078616 and R01HL151855 to Dr. Rob Sladek.

Conflict of interest

The authors declare no potential conflict of interests.

APPENDIX

References

Allaire, J., & Chollet, F. (2021). keras: R interface to 'keras' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=keras` R package version 2.7.0.

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

Blanche, P., Kattan, M. W., & Gerds, T. A. (2019). The c-index is not proper for the evaluation of-year predicted risks. *Biostatistics*, *20*(2), 347–357.

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

Coradini, D., Daidone, M. G., Boracchi, P., Biganzoli, E., Oriana, S., Bresciani, G., … Marubini, E. (2000). Time-dependent relevance of steroid receptors in breast cancer. *Journal of clinical oncology*, *18*(14), 2702–2709.

Curtis, C., Shah, S. P., Chin, S.-F., Turashvili, G., Rueda, O. M., Dunning, M. J., … others (2012). The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, *486*(7403), 346–352.

Dispenzieri, A., Katzmann, J. A., Kyle, R. A., Larson, D. R., Therneau, T. M., Colby, C. L., … others (2012). Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population. In *Mayo clinic proceedings* (Vol. 87, pp. 517–523).

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

Gerds, T. A., & Kattan, M. W. (2021). *Medical risk prediction models: With ties to machine learning (1st ed.)*. Chapman and Hall/CRC. Retrieved from `https://doi.org/10.1201/9781138384484` R package version 2021.10.10.

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

Gulli, A., & Pal, S. (2017). *Deep learning with keras*. Packt Publishing Ltd.

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

Hughes-Hallett, D., Gleason, A. M., & McCallum, W. G. (2020). *Calculus: Single and multivariable*. John Wiley & Sons.

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

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

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

Kleinbaum, D. G., & Klein, M. (2012). *Survival analysis: a self-learning text* (Vol. 3). Springer.

Knaus, W. A., Harrell, F. E., Lynn, J., Goldman, L., Phillips, R. S., Connors, A. F., … others (1995). The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, *122*(3), 191–203.

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

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

Lu-Yao, G. L., Albertsen, P. C., Moore, D. F., Shih, W., Lin, Y., DiPaola, R. S., … others (2009). Outcomes of localized prostate cancer following conservative management. *Jama*, *302*(11), 1202–1209.

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

Pölsterl, S. (2020). scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *Journal of Machine Learning Research*, *21*(212), 1-6. Retrieved from `http://jmlr.org/papers/v21/20-729.html` Version 0.14.0.

R Core Team. (2021). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from

https://www.R-project.org/

Royston, P., & Parmar, M. K. (2002). 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), 2175–2197.

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

Sonabend, R. (2021). survivalmodels: Models for survival analysis [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=survivalmodels R package version 0.1.9.

Terry M. Therneau, & Patricia M. Grambsch. (2000). *Modeling survival data: Extending the Cox model*. New York: Springer. R package version 3.2-11.

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B., & Wei, L.-J. (2011). On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in medicine*, *30*(10), 1105–1117.

Ushey, K., Allaire, J., & Tang, Y. (2021). reticulate: Interface to 'python' [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=reticulate R package version 1.22.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace. Version 3.6.13.

Wang, H., & Li, G. (2019). Extreme learning machine cox model for high-dimensional survival analysis. *Statistics in medicine*, *38*(12), 2139–2156.

AUTHOR BIOGRAPHY

Jesse Islam A PhD Candidate in Quantitative life sciences, studying complex relationships in data in a survival context.