Published in final edited form as:

Stat Med. 2020 July 30; 39(17): 2339–2349. doi:10.1002/sim.8542.

Deep learning for survival outcomes

Jon Arni Steingrimsen, Samantha Morrison

Department of Biostatistics, Brown University, Providence, Rhode Island

Abstract

Deep learning is a class of machine learning algorithms that are popular for building risk prediction models. When observations are censored, the outcomes are only partially observed and standard deep learning algorithms cannot be directly applied. We develop a new class of deep learning algorithms for outcomes that are potentially censored. To account for censoring, the unobservable loss function used in the absence of censoring is replaced by a censoring unbiased transformation. The resulting class of algorithms can be used to estimate both survival probabilities and restricted mean survival. We show how the deep learning algorithms can be implemented by adapting software for uncensored data by using a form of response transformation. We provide comparisons of the proposed deep learning algorithms to existing risk prediction algorithms for predicting survival probabilities and restricted mean survival through both simulated datasets and analysis of data from breast cancer patients.

Keywords

censoring unbiased transformations; doubly robust estimation; L_2 -loss; machine learning; restricted mean survival; risk estimation

1 | INTRODUCTION

Prediction models built using deep learning algorithms have had great success in many application areas including natural language processing,¹ speech recognition,² and image recognition.³ Deep learning algorithms are constructed using a sequence of layers each consisting of a nonlinear activation function that depends on an unknown vector of weights that is estimated by minimizing a loss function often subject to some regularization.

In medical studies, the outcome of interest is commonly time to a specific event such as time until death or disease progression. Such outcomes are frequently subject to right-censoring, which occurs when a subject drops out from the study, dies from other causes, or the study ends before the participant experiences the event of interest. The main difficulty of adapting

Correspondence: Jon Arni Steingrimsen, Department of Biostatistics, Brown University, Providence, RI 02903, USA.
jon_steingrimsen@brown.edu.

DATA ACCESSIBILITY

All data used in this article is publicly available from the R package *penalized* and the code used for analysis is publicly available from github.com/jonsteingrimsen/CensoringDL.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

the deep learning algorithm to right censored outcomes is that the full data loss used in the absence of censoring cannot be calculated.

To overcome that challenge, Liao et al⁴ and Ranganath et al⁵ proposed a deep learning algorithm where the loss function assumes a Weibull distributed failure time. Building on previous work,⁶ Katzman et al⁷ proposed a deep learning algorithm to estimate the functional form of the covariates in an underlying proportional hazard model using a loss function based on the partial likelihood of a proportional hazard model. Mobadersany et al⁸ used the algorithm for survival prediction based on digital pathology images and Li et al⁹ used a similar algorithm to predict overall survival for rectal cancer patients. Finally, several authors have proposed modeling the survival time using discrete distributions.^{10,11}

With the exception of the methods that assume discrete survival times, all of the aforementioned references build a prediction model where the survival time is assumed to have a parametric form or the loss function used relies on the proportional hazard assumption. Furthermore, the loss functions used do not reduce to the L_2 loss in the absence of censoring. As the L_2 loss is the most commonly used loss for deep learning algorithms with uncensored continuous outcomes, this results in a gap between what is done in the absence and presence of censoring.

Censoring unbiased loss functions¹² (CULs) are loss functions that can be calculated in the presence of censoring and satisfy: (i) they are unbiased estimators for the full data risk that would be used when there is no censoring and (ii) they reduce to the corresponding full data loss in the absence of censoring. Two important CULs are the doubly robust CUL and the Buckley-James CUL. This manuscript develops a class of deep learning algorithms for censored outcomes, referred to as censoring unbiased deep learning (CUDL), where the unobservable full data loss is replaced by the CUL functions.

Estimators of survival probabilities are the most common target parameters for censored risk prediction models. Estimators based on restricted mean survival are promising alternatives that have received substantial attention.¹³ We show how the full data loss for CUDLs can be selected to estimate both survival probabilities and restricted mean survival. To the best of our knowledge, this is the first deep learning algorithm that directly estimates restricted mean survival. Furthermore, we show how the CUDL algorithm can be implemented using software for fully observed continuous outcomes using a form of response transformation. This has important practical implications as it allows the algorithms developed to be implemented using standard software for fully observed outcomes.

Section 2 reviews the deep learning algorithm when there is no censoring. Section 3 discusses loss estimation for time-to-event outcomes. Section 4 introduces the CUDL algorithms and shows how different choices of full data loss functions result in deep learning algorithms that estimate both survival probabilities and restricted mean survival. Section 5 shows how the CUDL algorithms with the full data loss as the L_2 loss can be implemented using software for fully observed data. Sections 6 and 7 discuss implementation of the CUDL algorithms and evaluate the performance of the deep learning algorithms using

simulations and by analyzing data on breast cancer patients, respectively. A Supplementary Web Appendix contains additional simulation results and proofs.

2 | DEEP LEARNING FOR FULLY OBSERVED OUTCOMES

As it forms the building block for extensions to censored outcomes, we start by briefly describing the deep learning algorithm for uncensored outcomes. In the absence of censoring, the dataset is assumed to consist of a positive continuous failure time $T \in \mathbb{R}^+$ and a covariate vector $W \in \mathbb{R}^p$. We assume that the outcome is possibly transformed using a mono-tone function $h : \mathbb{R}^+ \rightarrow \mathbb{R}$ (e.g., $h(u) = u$ or $h(u) = \log(u)$). When there is no censoring, the data is $\mathcal{F} = \{(W_i, h(T_i)); i = 1, \dots, n\}$, and $\mathcal{F}$ is referred to as the fully observed data.

An important component of the deep learning algorithm is specification of a loss function.¹⁴ A loss function $L(h(T), f(W))$ measures the discrepancy between a prediction $f(W)$ and an outcome $h(T)$. Examples of loss functions commonly used in connection with a continuous outcome are the L_2 loss $(h(T) - f(W))^2$ and the L_1 loss $|h(T) - f(W)|$.

In this manuscript, we focus on the following deep learning algorithm based on feedforward networks. For a predetermined loss function, the full data deep learning algorithm is defined by the following steps:

1. *Setup the structure of the risk prediction model:* This step creates a risk prediction model using a series of functions (commonly referred to as layers) that are combined into a risk prediction model using the composition operator. More precisely, fix the number of layers K and for each layer $k \in \{1, \dots, K\}$ pre-specify a nonlinear activation function $f_{\beta_k}^{(k)}(x)$. A commonly used activation function is the rectified linear unit activation function

$$f_{\beta_k}^{(k)}(x) = \max\left(\left(\gamma^{(k)}\right)^T x + \rho_k, 0\right), \text{ where } \beta_k = \left(\text{vec}(\gamma^{(k)})^T, \rho_k^T\right)^T.$$

Here, for a matrix A , $\text{vec}(A)$ denotes the vectorization of the matrix A and the maximum used in the definition of the rectified linear unit activation function is an elementwise maximum. The input x into $f_{\beta_k}^{(k)}(x)$ is a vector of length d_{k-1} , $\gamma^{(k)}$ is a matrix of

dimensions $d_{k-1} \times d_k$ and the bias term ρ_k is a vector of length d_k (with $d_0 = p$ and $d_K = 1$). The final risk prediction model output by the architecture is

$$f_{\beta}(W) = f_{\beta_K}^{(K)} \circ f_{\beta_{K-1}}^{(K-1)} \circ \dots \circ f_{\beta_1}^{(1)}(W).$$

Here, $\beta = (\beta_1^T, \dots, \beta_K^T)^T$ is the vector of unknown weights that need to be estimated. We refer to the last layer, corresponding to $f_{\beta_K}^{(K)}$, as the output layer and all other layers as the hidden layers.

2. *Estimate the weight vector:* For a prespecified loss function $L(h(T), f(W))$ and a fixed value of the scalar penalization parameter η , the vector of weights β is estimated by minimizing the empirical penalized loss function

$$\frac{1}{n} \sum_{i=1}^n L(h(T_i), f_{\beta}(W_i)) + \eta \|\beta\|^2. \quad (1)$$

Here, $\|\beta\|^2 = \sum_{i=1}^q |\beta_i|^2$ and q is the length of the weight vector.

3. *Use cross-validation to select the penalization parameter η from a predetermined sequence $\eta_1, \dots, \eta_M$.* Randomly split the dataset into D disjoint sets $K_1, \dots, K_D$. For fixed $l \in \{1, \dots, D\}$ and $m \in \{1, \dots, M\}$, define $\hat{\beta}_{\eta_m}^l(W)$ as the vector of weights estimated by minimizing (1) using the penalization parameter η_m calculated using the data $\mathcal{F}/K_l$. Let $A_{i,l}$ be equal to one if observation i falls in dataset K_l and zero otherwise. The cross-validation error corresponding to η_m is defined as

$$\alpha(m) = \sum_{l=1}^D \sum_{i=1}^n A_{i,l} L(h(T_i), f_{\hat{\beta}_{\eta_m}^l}(W_i)). \quad (2)$$

The final value of η is η_{m^*} where $m^* = \arg \min_{m \in \{1, \dots, M\}} \alpha(m)$.

4. *Create the final prediction model:* The final prediction model is $f_{\hat{\beta}_{\eta_{m^*}}}(W)$, where $\hat{\beta}_{\eta_{m^*}}$ is the value of β that minimizes (1) with the penalization parameter set to η_{m^*} .

The population parameter that the full data deep learning algorithm estimates is the minimizer of the expected loss (risk) used in the algorithm. For the L_2 risk, the population parameter that the full data deep learning algorithm estimates is the conditional mean $E[h(T)|W]$. When computational complexity of the algorithm is an important consideration, a single split into a training and a tuning set is commonly used to select the value of the penalization parameter.

3 | RISK ESTIMATION WITH CENSORED DATA

In the presence of censoring, denoted C , the failure time is sometimes only partially observed. The data on observation i is assumed to be $O_i = (\tilde{T}_i = \min(T_i, C_i), \Delta_i = I(T_i \leq C_i), W_i)$. Here, $I(\cdot)$ is the indicator function. We refer to $\tilde{T}$ as the observed time and Δ as the failure time indicator. We assume that the observed dataset $\mathcal{O} = \{O_i; i = 1, \dots, n\}$ consists of n independent and identically distributed observations.

Define $S_0(u|W) = P(T > u|W)$ and $G_0(u|W) = P(C > u|W)$ as the conditional survival functions for T and C , respectively. We assume that C is continuous and that C is independent of T conditioned on W (non-informative censoring). We also make the positivity assumption $G_0(\tilde{T}|W) \geq \epsilon > 0$ for some $\epsilon > 0$. The full data loss function $L(h(T), \mathcal{R}(W))$ cannot be calculated if the failure time is censored. Replacing the unobservable full data loss with a loss function that can be calculated in the presence of censoring is the main difficulty in extending the deep learning algorithm to survival data.

An estimator is said to be an observed data estimator if it is a function of $\mathcal{O}$. Observed data estimators that are unbiased estimators of $\mathcal{R}(\beta) = E[L(h(T), \beta(W))]$ are referred to as censoring unbiased estimators for $\mathcal{R}(\beta)$. We now briefly describe two censoring unbiased estimators for $\mathcal{R}(\beta)$ that both reduce to the full data loss when there is no censoring.

The doubly robust loss function is given by

$$L_{DR}(\mathcal{O}, \beta; G_0, S_0) = \frac{1}{n} \sum_{i=1}^n \frac{\Delta_i L(h(T_i), \beta(W_i))}{G_0(\tilde{T}_i | W_i)} + \frac{1}{n} \left(\sum_{i=1}^n \frac{(1 - \Delta_i) m_L(\tilde{T}_i, W_i, \beta; S_0)}{G_0(\tilde{T}_i | W_i)} - \int_0^{\tilde{T}_i} \frac{m_L(u, W_i, \beta, S_0)}{G_0(u | W_i)} d\Lambda_{G_0}(u | W_i) \right). \quad (3)$$

where for a survival curve S

$$m_L(u, w, \beta; S) = E_S[L(h(T), \beta(W)) | T \geq u, W = w] = - \int_u^\infty \frac{L(h(t), \beta(w))}{S(u | w)} ds(t | w). \quad (4)$$

Here, $\Lambda_G(u | W) = - \int_0^u dG(t | W) / G(u | W)$ is the cumulative hazard function. The first term of the loss function (3) only uses uncensored outcomes, and the second term, commonly referred to as the augmentation term, is constructed to use information from censored responses in an efficient manner.¹⁵

Implementation of the doubly robust loss function (3) relies on estimating $G_0(\cdot | \cdot)$ and $S_0(\cdot | \cdot)$. Plugging observed data estimators $\hat{G}(\cdot | \cdot)$ and $\hat{S}(\cdot | \cdot)$ into (3) results in the empirical loss

$$L_{DR}(\mathcal{O}, \beta; \hat{G}, \hat{S}) = \frac{1}{n} \sum_{i=1}^n \frac{\Delta_i L(h(T_i), \beta(W_i))}{\hat{G}(\tilde{T}_i | W_i)} + \frac{1}{n} \left(\sum_{i=1}^n \frac{(1 - \Delta_i) m_L(\tilde{T}_i, W_i, \beta; \hat{S})}{\hat{G}(\tilde{T}_i | W_i)} - \int_0^{\tilde{T}_i} \frac{m_L(u, W_i, \beta, \hat{S})}{\hat{G}(u | W_i)} d\Lambda_{\hat{G}}(u | W_i) \right). \quad (5)$$

The loss function $L_{DR}(\mathcal{O}, \beta; \hat{G}, \hat{S})$ is doubly robust in that it is a consistent estimator for $\mathcal{R}(\beta)$ if one of the models for $T | W$ or $C | W$ is correctly specified but not necessarily both. For that reason, $L_{DR}(\mathcal{O}, \beta; \hat{G}, \hat{S})$ is referred to as the empirical doubly robust loss.

A related class of simple and intuitive observed data estimators are the Buckley-James estimators.^{16,17} In the context of risk estimation, the Buckley-James estimator for $\mathcal{R}(\beta)$ is given by

$$L_{BJ}(\mathcal{O}, \beta; \hat{S}) = \frac{1}{n} \sum_{i=1}^n \left(\Delta_i L(h(T_i), \beta(W_i)) + (1 - \Delta_i) m_L(C_i, W_i, \beta; \hat{S}) \right). \quad (6)$$

The Buckley-James loss requires a consistent estimator for $S_0(\cdot | \cdot)$ in order to consistently estimate $\mathcal{R}(\beta)$. The Buckley-James estimator has the optimality property that $L_{BJ}(\mathcal{O}, \beta; S_0)$ is the function of the observed data that minimizes $E[(\mathcal{R}(\beta) - f(\mathcal{O}))^2]$ for any observed data function $f(\mathcal{O})$.¹⁸ As $L_{BJ}(\mathcal{O}, \beta; \hat{S}) = L_{DR}(\mathcal{O}, \beta; \hat{S}, \hat{G} = 1)$, the Buckley-James loss is equivalent to the doubly robust loss using the (incorrect) model specification $\hat{G}(t|w)$ for all pairs (t, w) .

4 | CENSORING UNBIASED DEEP LEARNING

Replacing the full data loss function in the full data deep learning algorithm by either of the CULs results in a prediction model that can be implemented using the censored data $\mathcal{O}$. The algorithm obtained by replacing the full data loss by $L_{DR}(\mathcal{O}, \beta; \hat{G}, \hat{S})$ or $L_{BJ}(\mathcal{O}, \beta; \hat{S})$ are, respectively, referred to as the doubly robust and Buckley-James deep learning algorithms. Collectively we refer to these deep learning algorithms as CUDL, and as L_2 CUDL when implemented using the full data loss as the L_2 loss.

4.1 | Algorithms estimating $P(T \geq t | W)$

Survival probabilities of the form $P(T \geq t | W)$ for a fixed time-point t are common target parameters for time-to-event data. The Brier risk $E[(I(T \geq t) - \beta(W))^2]$ induces $P(T \geq t | W)$ as the target parameter as it is the function (W) that minimizes the Brier risk.

For a time-point t , define the modified dataset

$$\mathcal{O}(t) = \{(\tilde{T}_i(t) = \min(T_i, C_i, t), \Delta_i(t) = I(\min(T_i, t) \leq C_i), W_i); i = 1, \dots, n\}.$$

Applying the developments from Section 3 with the full data loss as the Brier loss $(I(T \geq t) - \beta(W))^2$ gives the doubly robust Brier loss

$$\begin{aligned} L_{DR}(\mathcal{O}(t), \beta; \hat{G}, \hat{S}) &= \frac{1}{n} \sum_{i=1}^n \frac{\Delta_i(t)(I(\tilde{T}_i(t) \geq t) - \beta(W_i))^2}{\hat{G}(\tilde{T}_i(t) | W_i)} \\ &\quad + \frac{1}{n} \sum_{i=1}^n \frac{(1 - \Delta_i(t)) m_{L_2, t}(\tilde{T}_i(t), W_i, \beta, \hat{S})}{\hat{G}(\tilde{T}_i(t) | W_i)} \\ &\quad - \int_0^{T_i(t)} \frac{m_{L_2, t}(u, W_i, \beta; \hat{S}) d\hat{\Lambda}_G(u | W_i)}{\hat{G}(u | W_i)}, \end{aligned} \quad (7)$$

where $m_{L_2, t}(u, W_i, \beta; \hat{S}) = E_S[(I(\tilde{T} \geq t) - \beta(W))^2 | T \geq u, W]$.

Following the developments in Section 3, the empirical Buckley-James Brier loss is given by

$$L_{BJ, t}(\mathcal{O}(t), \beta; \hat{S}) = \frac{1}{n} \sum_{i=1}^n (\Delta_i(t)(I(\tilde{T}_i \geq t) - \beta(W_i))^2 + (1 - \Delta_i(t))m_{L_2, t}(C_i, W_i, \beta; \hat{S})).$$

Incorporating the doubly-robust or Buckley-James Brier loss functions into the CUDL algorithm results in a prediction model for $P(T \leq t|W)$. We refer to the CUDL algorithms with the Brier loss selected to be the full data loss as the Brier CUDL algorithms.

4.2 | Algorithms estimating restricted mean survival

A popular target estimator for the full data deep learning algorithm for continuous outcomes is $E[T|W]$. Setting $h(u) = u$ and selecting the L_2 loss function to be the full data loss function in the full data deep learning algorithm results in an estimator for $E[T|W]$. But, estimating the mean with censored outcomes requires strong assumptions.¹⁹ A popular alternative for censored outcomes is to estimate the restricted mean survival $E[\min(T, \tau)|W]$ for some prespecified time-point τ . Selecting the full data loss as the L_2 loss and fitting the CUDL algorithms on the modified dataset $\mathcal{O}(\tau)$ results in an estimator for the restricted mean survival $E[\min(T, \tau)|W]$. We refer to the CUDL algorithm estimating $E[\min(T, \tau)|W]$ as the restricted mean survival CUDL algorithm.

5 | IMPLEMENTATION OF THE L_2 CUDL ALGORITHMS

This section shows how a form of response transformation can be used to implement the doubly robust and Buckley-James CUDL algorithms described in Sections 4.1 and 4.2. Specifically, these algorithms are implemented with deep learning software by fitting the full data deep learning algorithm described in Section 2 with an L_2 loss using a form of response transformation.

Both the restricted mean survival and Brier CUDL algorithms use an L_2 full data loss of the form $(h(T) - \beta(W))^2$ with $h(T) = \min(T, \tau)$ fit on the dataset $\mathcal{O}(\tau)$ and $h(T) = I(T \leq t)$ fit on the dataset (t) , respectively. Hence, both algorithms can be implemented using the form of response transformation described as follows.

For $k = 0, 1, 2$, define

$$A_{ki}(G) = \frac{\Delta_i h(\tilde{T}_i)^k}{G(\tilde{T}_i|W_i)}, \quad B_{ki}(G, S) = \frac{(1 - \Delta_i)m_k(\tilde{T}_i, W_i; S)}{G(\tilde{T}_i|W_i)},$$

and

$$C_{ki}(G, S) = \int_0^{T_i} \frac{m_k(u, W_i; S) d\Lambda_G(u|W_i)}{G(u|W_i)}$$

Here,

$$m_k(t, w; S) = E_S[h^k(T) | T \geq t, W = w] = -[S(t|w)]^{-1} \int_t^\infty [h(u)]^k dS(u, w)$$

for $k = 1, 2$ and $m_0(t, w; S) = 1$ for all pairs (t, w) . Define the response transformation

$$D(O_i; G, S) = A_{1i}(G) + B_{1i}(G, S) - C_{1i}(G, S).$$

As $E[D(O, G_0, S)] = E[D(O, G, S_0)] = E[h(T)]$, $D(O, G, S)$ is a censoring unbiased transformation for $E[h(T)]$ if at least one of the models for $T|W$ or $C|W$ is correctly specified. Define the response transformed L_2 loss as

$$L_2^*(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^n (D(O_i; G, S) - \beta(W_i))^2.$$

The response transformed L_2 loss is the L_2 loss using the censoring unbiased outcome transformation $D(O, G, S)$ as the outcome.

The doubly robust L_2 loss, denoted by $L_{DR}^{(2)}(\mathcal{O}, \beta; \hat{G}, \hat{S})$, is the doubly robust loss given by Equation (5) with $L(h(T), \beta(W)) = (h(T) - \beta(W))^2$. Theorem 1 shows that to implement CUDL algorithms using $L_{DR}^{(2)}(\mathcal{O}, \beta; \hat{G}, \hat{S})$, we can equivalently implement full data deep learning algorithms with the response transformed L_2 loss on the dataset $\{(D(O_i; \hat{G}, \hat{S}), W_i); i = 1, \dots, n\}$. A proof of the theorem is presented in Supplementary Web Appendix S.3.

Theorem 1.

The prediction model created using the CUDL algorithm with the loss function $L_{DR}^{(2)}(\mathcal{O}, \beta; G, S)$ is identical to the prediction model built using the full data deep learning algorithm implemented using the loss function $L_2^*(\mathcal{O}, \beta; G, S)$.

Theorem 1 is general enough to allow for $G(t|W = w) = 1$ for all pairs (t, w) , and as $L_{DR}(\mathcal{O}, \beta; G = 1, S) = L_{BJ}(\mathcal{O}, \beta; S)$ the result also holds for the Buckley-James loss. The main practical utility of Theorem 1 is that it allows the L_2 doubly robust and Buckley-James CUDL algorithms to be implemented using the road-map presented in Algorithm 1.

Algorithm 1. Implementing the CUDL algorithms using software for fully observed outcomes

- 1: Use the observed data $\hat{S}(\cdot | \cdot)$ and/or $\hat{G}(\cdot | \cdot)$.
 - 2: Create the response transformation $D(O_i; \hat{G}, \hat{S}), i = 1, \dots, n$.
 - 3: Run software implementing the L_2 full data deep learning algorithm on the dataset $\{(D(O_i; \hat{G}, \hat{S}), W_i); i = 1, \dots, n\}$.
-

Within the R software environment, the fully observed L_2 deep learning algorithm can be implemented using the Keras interface to R. Keras is a high level application programming interface that incorporates various types of backend engines such as Tensorflow, CNTK, or Theano to train deep learning models. There is a large literature on optimization techniques to estimate the weight vector for fully observed outcomes. The road-map described by Algorithm 1 allows easy implementation of the CUDL algorithms using optimization procedures available for fully observed outcomes and the L_2 loss. All the simulations and analysis presented in Sections 6 and 7 are implemented using the road-map given by Algorithm 1 with the Keras interface.

6 | SIMULATIONS

6.1 | Architecture and implementation choices

Implementation of the CUDL algorithms requires specifying the full data loss, the hidden and output layers, and the models for $S_0(\cdot|\cdot)$ and/or $G_0(\cdot|\cdot)$. The CUDL algorithms used in the simulations have one hidden layer ($K = 2$). The hidden layer consists of a rectified linear unit activation function with $d_1 = 15$ output units. That is, $f_{\beta_1}^{(1)} : \mathbb{R}^p \rightarrow \mathbb{R}^{d_1}$ with

$f_{\beta_1}^{(1)}(x) = \max(\gamma_1^T x + \rho_1, 0)$. Here, γ_1 is a matrix of dimension $p \times d_1$ and the bias ρ_1 is a vector of length d_1 . The maximum in the above equation is an elementwise maximum. The vector of weights β_1 is given by $\beta_1 = (\text{vec}(\gamma_1)^T, \rho_1^T)^T$.

For the Brier CUDL, the output layer $f_{\beta_2}^{(2)} : \mathbb{R}^{d_1} \rightarrow [0, 1]$ uses the sigmoid activation function

$f_{\beta_2}^{(2)}(x) = \left(1 + e^{-\gamma_2^T x - \rho_2}\right)^{-1}$, where γ_2 is a vector of length d_1 and ρ_2 is a scalar. The second vector of weights is $\beta_2 = (\gamma_2^T, \rho_2)^T$. The sigmoid activation function ensures that the final prediction falls in the interval $[0, 1]$, respecting the natural boundary of the target parameter $P(T \leq t|W)$. Section S.2.1 in Supplementary Web Appendix shows simulation results when the number of hidden layers is increased to 3 and 5. The restricted mean survival CUDL algorithms use the same architecture apart from the output layer being a rectified linear unit activation function instead of the sigmoid activation function (i.e., $f_{\beta_2}^{(2)}(x) = \max(\gamma_2^T x + \rho_2, 0)$).

This reflects that the target parameter is no longer a probability. For both CUDL algorithms, the length of the weight vector $\beta = (\beta_1^T, \beta_2^T)^T$ is $p * d_1 + 2 * d_1 + 1$.

We used 5-fold cross-validation to select the final penalization parameter from the sequence (0, 0.001, 0.01, 0.1). All covariates are standardized prior to fitting the CUDL algorithms. To solve the minimization problem required to estimate the weight vector, we use the rmsprop minimization procedure in the keras interface (version 2.2.4.1). We use default values for the tuning parameters except for setting the epochs parameter to 100 and we use 20% dropout.²⁰ Further details on implementation of the CUDL algorithms can be found in Supplementary Web Appendix S.1 and Supplementary Web Appendix S.3.1 provides the syntax for using the Keras interface. R code implementing the CUDL algorithm analysis presented in Section 7 is publicly available from github.com/jonsteingrimsdottir/CensoringDL.

6.2 | Simulation setup and results

We compare the two CUDL algorithms to a main effects Cox model, a penalized Cox model, random survival forests,²¹ and the deep surv algorithm.⁷ The deep surv algorithm is a deep learning algorithm where a loss function based on the partial likelihood of a Cox model is used to estimate the functional form of the covariates in a Cox model. We also compare the performance to random forests algorithms that use the Buckley-James and doubly robust loss functions described in Section 3.

The main effects Cox model is implemented using the `coxph` function in the `survival` package (version 2.42) in R. The penalized Cox model is implemented using the `cv.glmnet` function in R package `glmnet` (version 3.0). All default values of the tuning parameters were used for the penalized Cox model. This includes selecting the penalization parameter using 10 fold cross-validation. The random survival forest algorithm and the Buckley-James and doubly robust random forest algorithms are implemented using the `rfsrc` function from the `randomForest-SRC` package version 2.9.1²² with all tuning parameters set to their default values. Further details on the implementation of the Buckley-James and doubly robust random forest algorithms can be found in Supplementary Web Appendix S.1.

The deep surv algorithm is implemented using version 2.0.0 of the TFDeepSurv github python package. The TFDeepSurv package outputs a risk score defined as $\hat{k}(W)$ from the Cox model $\lambda(t) = \lambda_0(t)e^{\hat{k}(W)}$. We used the TFDeepSurv python package to calculate survival probabilities (implemented by estimating the baseline cumulative hazard using the Breslow estimator). For the TFDeepSurv implementation, all default tuning parameters were used except that the `hidden_layers_nodes` parameter was set to [2, 1]. This results in a 2 layer neural network, with a hidden layer of width 2 and an output layer of width 1. The number of layers and the size of the layer impact model training since more layers of larger width can better explore the covariate space but can also be prone to overfitting. Thus, choosing tuning parameters requires balancing complexity of the model with avoiding overfitting to the training data. We changed the `hidden_layers_nodes` parameter as the performance was substantially improved compared to using the default parameters (3 layer neural network); the CUDL algorithms also showed worse performance as the number of layers increased (See Figure S.1 in Supplementary Web Appendix).

We use the following simulation settings to compare the performance of the eight algorithms.

Setting 1.—The covariate vector is simulated from a 30-dimensional multivariate normal distribution with mean zero and covariance matrix with element (i, j) equal to $0.5^{|i-j|}$. The failure time distribution is exponential with mean $e^{0.1 \sum_{j=1}^{10} W^{(j)}}$, where $W^{(j)}$ is the j -th component of W . The censoring distribution is exponential with mean 1.14, which results in approximately 47 % censoring. The training set consists of 1000 i.i.d. draws from the joint distribution of $(\tilde{T}, \Delta, W)$. In this setting, the proportional hazard assumption is satisfied.

Setting 2.—The covariate vector is simulated from a 30 dimensional multivariate normal distribution with mean zero and covariance matrix with element (i, j) equal to $0.5^{|i-j|}$. The

failure time is simulated from a gamma distribution with shape parameter $0.5 + 0.3 \left| \sum_{j=1}^{15} W^{(j)} \right|$ and scale parameter equal to 2. The censoring times are uniformly distributed on the interval $[0, 15]$, which results in approximately 18% censoring. The training set consists of 1000 i.i.d. draws from the joint distribution of $(\tilde{T}, \Delta, W)$. In this setting, the proportional hazard assumption is violated.

We compare the algorithms both when predicting $P(T \leq t | W)$ for a pre-specified time-point t and when predicting restricted mean survival $E[\min(T, \tau) | W]$ for a prespecified time-point τ . For both simulation settings, the timepoint t is selected as the median of the marginal failure time distribution and τ is set to the 85th quantile of the marginal observed time distribution.

The Cox models and the random survival forest algorithm estimate the survival curve $S_0(t | W)$. The deep surv algorithm estimates a risk score, which when combined with the Breslow estimator for the cumulative baseline hazard can be used to estimate the survival curve $S_0(t | W)$. To estimate the restricted mean survival for these algorithms we calculate $\hat{S}(t | W)$ and use the formula $\hat{E}[\min(T, \tau) | W = w] = - \int_0^\infty \min(t, \tau) d\hat{S}(t | W)$.

The doubly robust and Buckley-James CUDL algorithms are implemented using the doubly robust and Buckley-James loss functions. To estimate $P(T \geq t | W)$ the CUDL algorithms use the full data loss as the Brier loss and to estimate $E[\min(T, \tau) | W]$, the CUDL algorithms use the estimation procedure detailed in Section 4.2. For both simulation settings, the length of the weight vector β is 481.

When estimating the survival probability $P(T \geq t | W)$, each of the algorithms is fit on the training set and the resulting model fit is used to predict $P(T \geq t | W)$ on an independent test set consisting of 1000 observations simulated using the corresponding full data distribution. The evaluation measure used is the average L_2 distance between the predicted test set probability and the true probability $\frac{1}{1000} \sum_{i=1}^{1000} (P(T \geq t | W_i) - \hat{P}(T \geq t | W_i))^2$. When the target parameter is the restricted mean survival, the probability $P(T \geq t | W)$ is replaced by $E[\min(T, \tau) | W]$.

The results from 1000 simulations are shown in Figure 1. We see that both CUDL algorithms perform substantially better than the random survival forest, and the deep surv algorithms for both settings and both target parameters. Both CUDL algorithms perform better than the Buckley-James and the doubly robust random forests for both settings and both target parameters with the exception that the doubly robust random forest algorithm performs better when estimating restricted mean survival using simulation setting 1. When compared with the Cox models, the CUDL algorithms perform substantially better when the proportional hazard assumption is violated (Setting 2). For the setting where the proportional hazard assumption holds (Setting 1), the CUDL algorithms show similar performance to the Cox model when estimating survival probabilities but perform slightly worse than the Cox models when estimating restricted mean survival. The doubly robust and Buckley-James CUDL algorithms show similar performance for both settings and target parameters.

Supplementary Web Appendix S.2 presents additional simulation results for estimating $P(T \leq t|W)$ using modifications of settings 1 and 2.

- Figure S.4 shows the comparisons for the eight algorithms when the sample size is 250, 500, 1500, and 3000. The performance of all algorithms improves as sample size increases and the relative performance of the algorithms is similar to what is seen in Figure 1. In setting 2, the improvements of the CUDL algorithms compared with the Cox model become larger as the sample size is increased.
- Figure S.5 shows simulation results when the covariate dimension is increased to 100. The doubly robust and Buckley-James random forest algorithms show improved relative performance compared to the results presented in Figure 1.
- Figures S.6 and S.7 show the simulation results when the time-point t is set to the 25th and 75th quantile of the marginal failure time distribution, respectively. For the 75th quantile and setting 2, the relative performance of the random forest algorithms is better than for the 50th quantiles. For both the 25th and 75th quantiles, the deep surv algorithm performs substantially worse than the CUDL algorithms for setting 1 and similarly for setting 2. Otherwise, the relative performance of the algorithms is similar to what is shown in Figure 1.
- Figure S.1 shows simulation results when the number of hidden layers is increased to 3 and 5. The results show that the CUDL algorithms with a single hidden layer perform better than the CUDL algorithms with 3 or 5 hidden layers.
- Supplementary Web Appendix S.2.2 examines the sensitivity of the CUDL algorithms to the choice of model for $S_0(t|W)$ and $G_0(t|W)$. We compare the prediction accuracy when $S_0(t|W)$ is estimated using random survival forests and three parametric accelerated failure time models and when $G_0(t|W)$ is estimated using a survival tree and a Kaplan-Meier estimator. The results suggest that the performance of the CUDL algorithms is not sensitive to the choice of models for $S_0(t|W)$ and $G_0(t|W)$.

7 | COMPARING PREDICTION ACCURACY USING THE NETHERLANDS 70 GENE SIGNATURE DATA

The Netherlands Cancer Institute 70 gene signature dataset consists of data from 144 lymph node positive breast cancer patients. The dataset includes five risk factors (diameter of tumor, number of positive nodes, ER status, grade, and age) and 70 measures of gene expression.²³ We use the data to evaluate the prediction accuracy of the CUDL algorithms when predicting the probability of metastasis-free survival beyond a specific time-point (measured in months). Patients who were alive at the end of study, developed a second primary cancer, had recurrence of regional or local disease, or died from other causes than breast cancer are considered censored. The censoring rate is 67%. The dataset is publicly available from the R package `penalized`.

We compare the prediction accuracy of the doubly robust and Buckley-James CUDL algorithms with a penalized Cox model, the random survival forest algorithm, the deep surv

algorithm, and the doubly robust and Buckley-James random forest algorithms. The main effects Cox model is not included as the algorithm failed to converge and had low prediction accuracy. All implementation choices needed to fully define the algorithms are as described in Section 6. Due to poor performance of the deep surv algorithm using the same tuning parameters as described in Section 6, the results presented use the tanh activation function and set the *hidden_layers_nodes* argument to [5, 1] (2 layer neural network, with a hidden layer of width 5 and an output layer of width 1). For comparison, we also implemented the CUDL algorithms using the tanh activation function (with all other implementation choices being as before).

The nine algorithms estimate $P(T \leq t|W)$ for a sequence of fixed time-points t and we compare the prediction accuracy using the censored data Brier score,²⁴ an MSE type measure. To calculate the censored data Brier score we use five fold cross-validation where the cross-validation is done such that all the cross-validation sets have approximately equal censoring rate.

Figure 2 shows the median of the censored data Brier score as a function of t across 100 different splits into cross-validation sets. For all time-points t , the CUDL algorithms and the deep surv algorithm have better or similar prediction accuracy compared to the penalized Cox model and the random forest algorithms. Overall, the CUDL algorithms that use the tanh activation function and the deep surv algorithm have the best performance. The doubly robust and Buckley-James algorithms show similar prediction accuracy.

8 | DISCUSSION

We developed a class of deep learning algorithms for time-to-event outcomes by replacing the unobservable full data loss used in the absence of censoring by a CUL. We show how the full data loss can be selected to estimate both survival probabilities and restricted mean survival. Furthermore, we show that the doubly robust and Buckley-James deep learning algorithms can be implemented using standard software for fully observed outcomes using a form of response transformation. The performance of the algorithms is evaluated both using simulations and by analyzing a dataset on breast cancer patients.

Implementation of the doubly robust and Buckley-James algorithm requires an estimator for $S_0(t|W)$ and/or $G_0(t|W)$. A theoretically interesting approach would be to utilize an iterative algorithm to update the estimator for $S_0(t|W)$ and/or $G_0(t|W)$ using the CUDL algorithm. More precisely, the first iteration implements the CUDL algorithm using some initial estimator for $S_0(t|W)$ and/or $G_0(t|W)$. The resulting CUDL algorithm is then used to estimate $S_0(t|W)$ and/or $G_0(t|W)$ and an updated CUDL algorithm is calculated with the updated estimators. This process is repeated with the updated estimators until convergence or for a fixed amount of iterations. However, in practice, this is very computationally intensive as the Brier CUDL algorithms estimate $P(T \leq t|W)$ for a fixed time-point t . The CUDL algorithm can be used to calculate an estimator for the whole survival curve by using the CUDL algorithm to calculate $P(T \leq t|W)$ setting t to each unique failure time in the dataset and assuming that the survival curve only jumps at observed failure times. This is a limitation of

the CUDL algorithm, as many algorithms such as the Cox model and the random survival forest algorithm estimate the whole survival curve in a single iteration.

There are several interesting future research directions arising from this work. Important extensions include extending the CUDL algorithms to more complex data structures such as competing risk and time-dependent covariates as well as extensions to convolution neural networks for imaging analysis. Furthermore, appropriately handling missing data both when the missingness mechanism is unknown and known (e.g., case-cohort studies) is of importance.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGMENTS

The authors thank Dr. Constantine Gatsonis and Dr. Alice Paul for helpful comments and discussions. This work was supported by grants U10CA180820 and U10CA180794 from the National Cancer Institute to the ECOG-ACRIN Cancer Research Group (Peter J. O'Dwyer, MD and Mitchell D. Schnall, MD, PhD, Group Co-Chairs). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. government. It was also supported by a Brown University Salomon award.

Funding information

National Cancer Institute, Grant/Award Numbers: U10CA180794, U10CA180820

REFERENCES

1. Goldberg Y A primer on neural network models for natural language processing. *J Artif Intell Res* 2016;57:345–420.
2. Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks. Paper presented at: Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing; 2013:6645–6649; IEEE.
3. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436. [PubMed: 26017442]
4. Liao L, Ahn H. Combining deep learning and survival analysis for asset health management. *Int J Prognost Health Manag*. 2016; 521:436–444.
5. Ranganath R, Perotte A, Elhadad N, Blei D. Deep survival analysis; 2016 arXiv preprint arXiv:1608.02158.
6. Faraggi D, Simon R. A neural network model for survival data. *Stat Med* 1995;14(1):73–82. [PubMed: 7701159]
7. Katzman JL, Shaham U, Cloninger A, Bates J, Jiang T, Kluger Y. DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Med Res Methodol* 2018;18(1):24. [PubMed: 29482517]
8. Mobadersany P, Yousefi S, Amgad M, et al. Predicting cancer outcomes from histology and genomics using convolutional networks. *Proc Natl Acad Sci* 2018;115:201717139.
9. Li H, Boimel P, Janopaul-Naylor J, et al. Deep convolutional neural networks for imaging data based survival analysis of rectal cancer; 2019 arXiv preprint arXiv:1901.01449.
10. Miskouridou X, Perotte A, Elhadad N, Ranganath R. Deep survival analysis: nonparametrics and missingness. Paper presented at: Proceedings of the Machine Learning for Healthcare Conference; 2018:244–256.
11. Gensheimer MF, Narasimhan B. A scalable discrete-time survival model for neural networks. *PeerJ* 2019;7:e6257. [PubMed: 30701130]

12. Steingrimsdottir JA, Diao L, Strawderman RL. Censoring unbiased regression trees and ensembles. *J Am Stat Assoc* 2019;114(525):370–383. [PubMed: 31190691]
13. Royston P, Parmar MK. The use of restricted mean survival time to estimate the treatment effect in randomized clinical trials when the proportional hazards assumption is in doubt. *Stat Med* 2011;30(19):2409–2421. [PubMed: 21611958]
14. Goodfellow I, Bengio Y, Courville A, Bengio Y. *Deep Learning*. Vol 1 Cambridge, MA: MIT Press; 2016.
15. Steingrimsdottir JA, Diao L, Molinaro AM, Strawderman RL. Doubly robust survival trees. *Stat Med* 2016;35(17–18):3595–3612. [PubMed: 27037609]
16. Buckley J, James I. Linear regression with censored data. *Biometrika* 1979;66(3):429–436.
17. Rubin D, van Laan DMJ. A doubly robust censoring unbiased transformation. *Int J Biostat* 2007;3(1):1–21.
18. Fan J, Gijbels I. Censored regression: local linear approximations and their applications. *J Am Stat Assoc* 1994;89(426):560–570.
19. Ding Y, Nan B. Estimating mean survival time: when is it possible? *Scand J Stat* 2015;42(2):397–413.
20. Dahl GE, Sainath TN, Hinton GE. Improving deep neural networks for LVCSR using rectified linear units and dropout. Paper presented at: Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing; 2013:8609–8613; IEEE.
21. Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS. Random survival forests. *Ann Appl Stat* 2008;2:841–860.
22. Ishwaran H, Kogalur UB. Random survival forests for R; 2007.
23. Van Veer LJ, Dai H, Vander Vijver MJ, et al. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*. 2002;415(6871):530–536. [PubMed: 11823860]
24. Graf E, Schmoor C, Sauerbrei W, Schumacher M. Assessment and comparison of prognostic classification schemes for survival data. *Stat Med* 1999;18(17–18):2529–2545. [PubMed: 10474158]

FIGURE 1.

Mean squared error for the eight different algorithms for both simulation settings described in Section 6.2. Lower values indicate better performance. The top row shows the performance when predicting $P(T \leq t|W)$ and the bottom row when estimating restricted mean survival $E[\min(T, \tau) | W]$. Cox and Cox Pen are a main effect Cox model and an L_1 penalized Cox model, respectively. RSF is the random survival forest algorithm and DS is the deep surv algorithm. DR DL and BJ DL are the doubly robust and Buckley-James deep learning algorithms, respectively and DR RF and BJ RF are the doubly robust and Buckley-James random forest algorithms, respectively

FIGURE 2.

Censored data Brier score²⁴ as a function of t when predicting $P(T \leq t | W)$ on the Netherlands Cancer Institute 70 gene signature data. Lower values indicate better performance. Cox Pen is a main effects L_1 penalized Cox model. RSF is the random survival forest algorithm, DS is the deep surv algorithm, and DR RF and BJ RF are the doubly robust and Buckley-James random forests algorithms. DR DL and BJ DL are the doubly robust and Buckley-James deep learning algorithms with the rectified linear unit activation function. DR DL Tanh and BJ DL Tanh are the doubly robust and Buckley-James deep learning algorithms with the tanh activation function