Journal of Machine Learning Research 25 (2024) 1-78 Submitted 6/22; Revised 1/24; Published 2/24

Survival Kernets: Scalable and Interpretable Deep Kernel Survival Analysis with an Accuracy Guarantee

George H. Chen GEORGECHEN@CMU.EDU
Heinz College of Information Systems and Public Policy
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Ji Zhu

Abstract

Kernel survival analysis models estimate individual survival distributions with the help of a kernel function, which measures the similarity between any two data points. Such a kernel function can be learned using deep kernel survival models. In this paper, we present a new deep kernel survival model called a *survival kernet*, which scales to large datasets in a manner that is amenable to model interpretation and also theoretical analysis. Specifically, the training data are partitioned into clusters based on a recently developed training set compression scheme for classification and regression called *kernel netting* that we extend to the survival analysis setting. At test time, each data point is represented as a weighted combination of these clusters, and each such cluster can be visualized. For a special case of survival kernets, we establish a finite-sample error bound on predicted survival distributions that is, up to a log factor, optimal. Whereas scalability at test time is achieved using the aforementioned kernel netting compression strategy, scalability during training is achieved by a warm-start procedure based on tree ensembles such as XGBOOST and a heuristic approach to accelerating neural architecture search. On four standard survival analysis datasets of varying sizes (up to roughly 3 million data points), we show that survival kernets are highly competitive compared to various baselines tested in terms of time-dependent concordance index. Our code is available at: `https://github.com/georgehc/survival-kernets`

Keywords: survival analysis, kernel methods, neural networks, scalability, interpretability

1. Introduction

Survival analysis is about modeling the amount of time that will elapse before a critical event of interest happens. Examples of such critical events include death, hospital readmission, disease relapse, device failure, or a convicted criminal reoffending. A key technical challenge in survival analysis is that typically when collecting training data, we cannot wait until the critical event happens for every training data point, such as waiting for everyone in a clinical study to be deceased. Thus, in our training data, we observe the time duration we aim to predict for only some but not all of the data points. This is in contrast to standard classification and regression problem settings in which the target label to be predicted is observed across all training data. Importantly, the data points that do not encounter the critical event could still provide valuable information and should not simply be discarded

from the data analysis. For instance, they may have specific characteristics that significantly delay when the critical event will happen.

In the last few decades, survival analysis datasets have dramatically grown in size, from hundreds of data points (e.g., the German Breast Cancer Study Group dataset (Schumacher et al., 1994)) to now millions (e.g., customer churn data from the music streaming service KKBOX[1]). We anticipate that in the years to come, survival analysis datasets larger than ones typically encountered today will become the norm. Meanwhile, we remark that many survival analysis problems are in high-stakes application domains such as healthcare. For such applications, it can be helpful for the survival analysis models used to be interpretable. With these large-scale high-stakes applications in mind, in this paper, we propose a new deep survival analysis model called a *survival kernet* that has all of the following properties:

- flexible (model is nonlinear and nonparametric, and achieves concordance indices nearly as high as or higher than the best baselines tested in our experiments)
- scalable (uses a compression technique to construct a test-time predictor that can handle large datasets such as the KKBOX dataset; also uses a scalable neural net warm-start initialization strategy)
- interpretable (any data point is represented by a weighted combination of a few clusters, each of which can be visualized—this is somewhat like how topic modeling for text describes each text document in terms of a few topics, each of which can be visualized)
- for a special case of the model, comes with a theoretical guarantee on prediction accuracy (a finite-sample error bound for predicted survival distributions)

In contrast, existing deep survival analysis models that have been developed typically are not easily interpretable (e.g., Ranganath et al. 2016; Fotso 2018; Chapfuwa et al. 2018; Giunchiglia et al. 2018; Katzman et al. 2018; Lee et al. 2018; Kvamme et al. 2019; Engelhard et al. 2020; Zhong et al. 2021; Danks and Yau 2022; Tang et al. 2022). Instead, to get a deep survival analysis model to be interpretable, Zhong et al. (2022), for instance, model a subset of features by a linear model (that is straightforward to interpret) and then model the rest of the features by an arbitrarily complex neural net that they do not aim to interpret. While this sort of interpretability could be valuable, it does not resolve the difficulty of interpreting the part of the model that actually uses the neural net. Meanwhile, Chapfuwa et al. (2020), Nagpal et al. (2021), and Manduchi et al. (2022) have developed deep survival models that are interpretable in the sense of representing data points in terms of clusters, but none of these models have accuracy guarantees. Somewhat like using clustering, neural survival topic models by Li et al. (2020) represents data in terms of "topics", where each topic corresponds to specific raw features being more probable and also to either higher or lower survival times; these models again lack accuracy guarantees. We are aware of only two existing deep survival analysis models that have accuracy guarantees (Zhong et al., 2021, 2022), both of which have only been tested on small datasets (the largest real dataset these authors consider has about two thousand data points) and the theoretical analyses for these models are limited to a special family of neural nets (involving ℓ_0 and sup-norm constraints) that are not used in practice. We summarize how our proposed survival kernet

1. `https://www.kaggle.com/c/kkbox-churn-prediction-challenge`

Table 1.1: Comparison of the proposed survival kernet model with some representative existing survival models.

Model	Nonlinear	No proportional hazards assumption	Designed to be interpretable*	Accuracy guarantee†	Comment on computational scalability‡
Cox (Cox, 1972)	✗	✗	✓	✓	SGD
XGBoost (Chen and Guestrin, 2016)	✓	✗§	✗	✗	various optimizations
DeepSurv (Katzman et al., 2018)	✓	✗	✗	✗	SGD
DeepHit (Lee et al., 2018)	✓	✓	✗	✗	SGD
Nnet-survival (Gensheimer and Narasimhan, 2019)	✓	✓	✗	✗	SGD
Cox-time (Kvamme et al., 2019)	✓	✓	✗	✗	SGD
Deep Extended Hazard (Zhong et al., 2021)	✓	✓¶	✗	✓	SGD
SODEN (Tang et al., 2022)	✓	✓	✗	✗	SGD
Neural survival topic models (Li et al., 2020)	✓	✓	✓	✗	SGD
Deep Cox mixtures (Nagpal et al., 2021)	✓	✓	✓	✗	uses spline fitting procedure with computation linear in training set size‖
Deep kernel survival analysis (Chen, 2020)	✓	✓	✓	✗	SGD but prediction for single point uses computation linear in training set size
Survival kernets (proposed)	✓	✓	✓	✓	SGD + tunable test-time compression + scalable warm-start

* For models designed to be interpretable, the notion of interpretability varies (the Cox model is linear, neural survival topic models decompose into a topic model and a survival model that each can be interpreted, Deep Cox mixtures and survival kernets represents data using clusters, and deep kernel survival analysis can explicitly indicate which training points contribute to each prediction and what their relative weights are)

† Accuracy guarantees for the deep extended hazard model and survival kernets are for special cases of these models

‡ For some models that can be trained using (minibatch) SGD, doing so is not actually theoretically justified but is straightforward to implement in practice

§ Specific to using Cox regression as the learning objective, which we do in our experiments

¶ The deep extended hazard model is a hybrid between a proportional hazards model and an accelerated failure time model so that it partially uses a proportional hazards assumption (but from training, the model could automatically decide to rely on the accelerated failure time component and not really use the proportional hazards component)

‖ Deep Cox mixtures uses an Expectation-Maximum (EM) training procedure that partly uses minibatches but every few minibatches looks at the entire training set in computing a survival curve (as a spline) per cluster

model compares to some representative existing models in Table 1.1; note that this table is partially based on a similar comparison table by Tang et al. (2022).

Our work builds on an existing model called deep kernel survival analysis (Chen, 2020). Unlike non-kernel-based deep survival models that have been developed, deep kernel survival analysis and a Bayesian variant by Wu et al. (2021) learn a kernel function that measures how similar any two data points are. To predict the survival distribution of a specific test point, these kernel-based methods use information from training points most similar to the test point according to the learned kernel function. The learned kernel function can help us probe a resulting survival model. For instance, these kernel functions can provide forecast evidence in terms of which training points contribute to a test point's prediction. As pointed out by Chen (2020), the kernel functions can also be used to construct statistically valid prediction intervals that are *relative* (e.g., the hospital length of stay of patients *similar to Alice* are within the interval [0.5, 2.5] days with probability at least 80%). Separate from these practical advantages, a kernel-based approach is also amenable to theoretical analysis, where we import proof techniques by Chen (2019) to analyze survival kernets.

A key obstacle to using kernel survival models in practice is that at test time, these models in principle depend on knowing the similarity between a test point and every training point. The computation involved in a naive implementation becomes impractical when the training dataset size grows large. In the Bayesian setting, this scalability problem has been

addressed but without guarantees in the survival analysis setting and also thus far without experiments on datasets with censoring (Wu et al., 2021).

Our main technical contribution in this paper is to show how to scale deep kernel survival analysis at test time to large datasets in a manner that not only yields an interpretable model but also achieves a finite-sample accuracy guarantee on predicted survival distributions. The resulting model is what we call a *survival kernet*. We achieve this test-time scalability by extending an existing data compression scheme (*kernel netting* by Kpotufe and Verma (2017) developed for classification and regression) to the survival analysis setting ("survival kernet" combines "survival analysis" and "kernel netting" into a short phrase). Kernel netting could be viewed as partitioning the training data into clusters and then representing any data point as a weighted combination of a few clusters. We show how to visualize such clusters using a strategy similar to that of Chapfuwa et al. (2020). We remark that Kpotufe and Verma did not consider interpretability or how to visualize clusters in their original kernel netting paper.

We next turn to training scalability. Here, minibatch gradient descent readily enables fitting a deep kernel survival model to large datasets for a specific neural net architecture. The challenge is that there could be many neural net architectures to try, making the overall training procedure computationally expensive. Our second contribution is in proposing a warm-start approach to deep kernel survival model training that drastically reduces the amount of computation needed when sweeping over neural net architecture choices. Our proposed warm-start strategy first learns a kernel function using a scalable tree ensemble such as XGBOOST (Chen and Guestrin, 2016). We then fit a neural net (trying different neural net architectures) to this learned tree ensemble kernel function before fine tuning using survival kernet training (at which point the neural net architecture is fixed). We call our warm-start approach TUNA (*Tree ensemble Under a Neural Approximation*). We remark that Chen (2020) had come up with an earlier tree ensemble initialization strategy but it does not scale to large datasets.

We demonstrate survival kernets with and without TUNA on four standard survival analysis datasets, three of which are healthcare-related on predicting time until death (with number of data points ranging from thousands to tens of thousands), and the fourth is the KKBOX dataset (millions of data points). Survival kernets with TUNA achieve accuracy scores nearly as high or higher than the best performing baselines that we tested. Meanwhile, using TUNA to accelerate training consistently results in higher accuracy models than not using TUNA and reduces overall training times by 17%–85% in our experiments (with time savings that are more dramatic on larger datasets). We show how to interpret survival kernet models trained on all four datasets we consider.

2. Background

We first review the standard right-censored survival analysis setup (model and prediction task) and deep kernel survival analysis (Chen, 2020). For the latter, we do not address the Bayesian formulation (e.g., Wu et al. 2021) as the machinery involved is a bit different and our theoretical analysis later is frequentist. For ease of exposition, we phrase terminology in terms of predicting time until death although in general, the critical event of interest need not be death.

Model Let $(X_1, Y_1, D_1), \ldots, (X_n, Y_n, D_n)$ denote the training data, where the i-th training point has feature vector $X_i \in \mathcal{X}$, observed nonnegative time duration $Y_i \geq 0$, and event indicator $D_i \in \{0, 1\}$; if $D_i = 1$, then Y_i is a time until death whereas if $D_i = 0$, then Y_i is a time until censoring (i.e., the i-th point's true time until death is at least Y_i). Each point (X_i, Y_i, D_i) is assumed to be generated i.i.d. as follows:

1. Sample feature vector $X_i \sim \mathbb{P}_{\mathsf{X}}$.
2. Sample nonnegative survival time $T_i \sim \mathbb{P}_{\mathsf{T}|\mathsf{X}=X_i}$.
3. Sample nonnegative censoring time $C_i \sim \mathbb{P}_{\mathsf{C}|\mathsf{X}=X_i}$.
4. If $T_i \leq C_i$ (death happens before censoring): set $Y_i = T_i$ and $D_i = 1$.
   Otherwise (death happens after censoring): set $Y_i = C_i$ and $D_i = 0$.

Distributions $\mathbb{P}_{\mathsf{X}}$, $\mathbb{P}_{\mathsf{T}|\mathsf{X}}$, and $\mathbb{P}_{\mathsf{C}|\mathsf{X}}$ are unknown to the learning method. We assume that the distribution $\mathbb{P}_{\mathsf{T}|\mathsf{X}=x}$ is for a continuous random variable with CDF $F_{\mathsf{T}|\mathsf{X}}(t|x)$ and PDF $f_{\mathsf{T}|\mathsf{X}}(t|x) = \frac{\partial}{\partial t} F_{\mathsf{T}|\mathsf{X}}(t|x)$.

Prediction task A standard prediction task is to estimate, for a test feature vector x, the conditional survival function

$$S(t|x) := \mathbb{P}(\text{time until death} > t \,|\, \text{feature vector} = x) = 1 - F_{\mathsf{T}|\mathsf{X}}(t|x),$$

which is defined for all $t \geq 0$. A closely related problem is to estimate the so-called hazard function

$$h(t|x) := -\frac{\partial}{\partial t} \log S(t|x) = -\frac{\frac{\partial}{\partial t} S(t|x)}{S(t|x)} = -\frac{\frac{\partial}{\partial t}[1 - F_{\mathsf{T}|\mathsf{X}}(t|x)]}{S(t|x)} = \frac{f_{\mathsf{T}|\mathsf{X}}(t|x)}{S(t|x)}, \tag{2.1}$$

which is (from the final expression) the instantaneous rate of death at time t given survival at least through time t for feature vector x. By how the hazard function is defined, $S(t|x) = \exp(-\int_0^t h(s|x)ds)$, so estimating $h(\cdot|x)$ yields an estimate of $S(\cdot|x)$.

Kernel estimators Let $\mathbb{K} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ denote a kernel function that measures how similar any two feature vectors are (a higher value means more similar). We explain how this function can be learned shortly in a neural net framework. For now, consider it to be pre-specified. Then we can estimate the hazard function as follows.

Hazard function estimator. Let $t_1 < t_2 < \cdots < t_m$ denote the unique times of death in the training data, and define $t_0 := 0$. Then a kernel predictor for a discretized version of the hazard function is, for time indices $\ell = 1, 2, \ldots, m$ and feature vector $x \in \mathcal{X}$,

$$\widehat{h}(\ell|x) := \frac{\sum_{j=1}^n \mathbb{K}(x, X_j) D_j \mathbb{1}\{Y_j = t_\ell\}}{\sum_{j=1}^n \mathbb{K}(x, X_j) \mathbb{1}\{Y_j > t_{\ell-1}\}}, \tag{2.2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function that is 1 when its argument is true and 0 otherwise. Note that $\widehat{h}(\ell|x)$ estimates the probability of dying at time t_ℓ conditioned on surviving beyond time $t_{\ell-1}$ for feature vector x. To see this, consider when $\mathbb{K}(x, X_j) = 1$ for all j, in which case the numerator counts the total number of deaths at time t_ℓ, and the denominator counts the total number of data points that survived beyond time $t_{\ell-1}$; adding kernel weights that are not necessarily always 1 weights the contribution of the j-th training point depending

on how similar test feature vector x is to X_j. As a corner case, in evaluating equation (2.2), if the numerator and denominator are both 0, we use the convention that $0/0 = 0$.

Survival function estimator. The hazard estimate $\widehat{h}(\cdot|x)$ can be used to estimate the conditional survival function $S(\cdot|x)$ by taking products of empirical probabilities of surviving from time 0 to t_1, t_1 to t_2, and so forth up to time t:

$$\widehat{S}(t|x) := \prod_{\ell=1}^{m} \big(1 - \widehat{h}(\ell|x)\big)^{\mathbb{1}\{t_\ell \le t\}}. \tag{2.3}$$

This kernel predictor is called the *conditional Kaplan-Meier estimator* (Beran, 1981) and has known finite-sample error bounds (Chen, 2019). For equation (2.3), the special case where $\mathbb{K}(x, x') = 1$ for all $x, x' \in \mathcal{X}$ yields the classical Kaplan-Meier estimator (Kaplan and Meier, 1958) that does not depend on feature vectors and is a population-level survival curve estimate:

$$\widehat{S}^{\mathrm{KM}}(t) := \prod_{\ell=1}^{m} \Big(1 - \frac{\sum_{j=1}^{n} D_j \mathbb{1}\{Y_j = t_\ell\}}{\sum_{j=1}^{n} \mathbb{1}\{Y_j > t_{\ell-1}\}}\Big)^{\mathbb{1}\{t_\ell \le t\}}. \tag{2.4}$$

Deep kernel survival analysis To automatically learn the kernel function $\mathbb{K}$, deep kernel survival analysis (Chen, 2020) parameterizes $\mathbb{K}$ in terms of a base neural net $\phi : \mathcal{X} \to \widetilde{\mathcal{X}}$ that maps a raw feature vector x to an embedding vector $\widetilde{x} = \phi(x) \in \widetilde{\mathcal{X}}$; throughout this paper we denote embedding vectors with tildes. We always assume that the embedding space $\widetilde{\mathcal{X}}$ is a subset of $\mathbb{R}^d$ and that kernel function $\mathbb{K}$ is of the form

$$\mathbb{K}(x, x'; \phi) = K(\rho(x, x'; \phi)), \;\; \rho(x, x'; \phi) = \|\phi(x) - \phi(x')\|_2, \tag{2.5}$$

where $K : [0, \infty) \to [0, \infty)$ is a nonincreasing function, and $\|\cdot\|_2$ denotes Euclidean distance. For example, a common choice is the Gaussian kernel $K(u) = \exp(-\frac{u^2}{2\sigma^2})$ where σ^2 is the user-specified variance hyperparameter. Learning $\mathbb{K}$ or distance function ρ amount to learning the base neural net ϕ. The architecture of ϕ is left for the user to specify, where standard strategies can be used. For example, when working with images, we can choose ϕ to be a convolutional neural net and when working with time series, ϕ can be a recurrent neural net.

To prevent overfitting during training, we replace the hazard estimate $\widehat{h}(\ell|x)$ in equation (2.2) with a "leave-one-out" training version, and in terms of notation, we now also emphasize the dependence on the base neural net ϕ:

$$\widehat{h}_{\mathrm{train}}(\ell|i; \phi) := \frac{\sum_{j=1 \text{ s.t. } j \neq i}^{n} \mathbb{K}(X_i, X_j; \phi) D_j \mathbb{1}\{Y_j = t_\ell\}}{\sum_{j=1 \text{ s.t. } j \neq i}^{n} \mathbb{K}(X_i, X_j; \phi) \mathbb{1}\{Y_j > t_{\ell-1}\}}. \tag{2.6}$$

In particular, the prediction for the i-th training point does not depend on the i-th training point's observed outcomes Y_i and D_i. Similarly, we can define a leave-one-out version $\widehat{S}^{\mathrm{train}}(t|x; \phi)$ of the survival function estimate $\widehat{S}(t|x)$ given in equation (2.3).

As for what loss function to minimize to learn the base neural net ϕ (and thus the kernel function $\mathbb{K}$ for use with the hazard or survival function estimators), one possibility is the

negative log likelihood loss by Brown (1975):

$$L_{\mathrm{NLL}}(\phi) := \frac{1}{n}\sum_{i=1}^{n}\Bigg(L_{\mathrm{BCE}}(i;\phi) + \underbrace{\sum_{\substack{\ell=1 \\ \text{s.t. } t_\ell < Y_i}}^{m} \log\frac{1}{1-\widehat{h}_{\mathrm{train}}(\ell|i;\phi)}}_{i\text{-th individual survives at times before } Y_i}\Bigg),$$

where $L_{\mathrm{BCE}}(i;\phi)$ is the binary cross-entropy loss

$$L_{\mathrm{BCE}}(i;\phi) := D_i \log\frac{1}{\widehat{h}_{\mathrm{train}}(\kappa(Y_i)|i;\phi)} + (1-D_i)\log\frac{1}{1-\widehat{h}_{\mathrm{train}}(\kappa(Y_i)|i;\phi)},$$

and $\kappa(Y_i)$ denotes the sorted time index (from $1, 2, \ldots, m$) that time Y_i corresponds to. Minimizing L_{NLL} via minibatch gradient descent yields the deep kernel survival analysis approach by Chen (2020).

Chen also discussed how to incorporate the DEEPHIT ranking loss term (Lee et al., 2018), which could be written

$$L_{\mathrm{rank}}(\phi) := \frac{1}{n^2}\sum_{\substack{i=1 \\ \text{s.t. } D_i=1}}^{n}\sum_{\substack{j\neq i \\ \text{s.t. } Y_j > Y_i}} \exp\Big(\frac{\widehat{S}_{\mathrm{train}}(Y_i|X_i;\phi) - \widehat{S}_{\mathrm{train}}(Y_i|X_j;\phi)}{\sigma_{\mathrm{rank}}}\Big),$$

where $\sigma_{\mathrm{rank}} > 0$ is a hyperparameter; for simplicity, we use a normalization factor of $1/n^2$, which is the same normalization factor used in the DEEPHIT implementation that is part of the now standard PYCOX software package (Kvamme et al., 2019). Our experiments later use the following overall deep kernel survival analysis (DKSA) loss term that trades off between Brown's loss L_{NLL} and the ranking loss L_{rank}:

$$L_{\mathrm{DKSA}}(\phi) := \eta L_{\mathrm{NLL}}(\phi) + (1-\eta)L_{\mathrm{rank}}(\phi), \tag{2.7}$$

where $\eta \in [0,1]$ is another hyperparameter.

Implementation remarks. As stated, the hazard estimator (2.2) as well as the conditional Kaplan-Meier estimator (2.3) are defined in terms of the unique observed times of death. In practice, Chen (2020) observed that discretizing time into time steps of equal size can sometimes yield more accurate survival predictions. This discretization acts as a form of regularization since we are essentially smoothing the resulting estimated hazard and conditional survival functions. Furthermore, note that the conditional Kaplan-Meier estimator is defined to be piecewise constant. However, in practice, interpolation improves test-time prediction accuracy. Chen's implementation of deep kernel survival analysis uses the constant density interpolation strategy by Kvamme and Borgan (2021).

Separately, during training, using an infinite-support kernel function such as a Gaussian kernel works well in practice with neural net frameworks to prevent vanishing gradients. If we use a kernel function with finite support (such as the box, triangle, or Epanechnikov kernels), and we initialize ϕ such that for each training point, no other training point is found to be similar enough as to have a nonzero kernel weight, then we would struggle to learn an improved embedding representation. As a concrete example, suppose that we use the box kernel $K(u) = \mathbb{1}\{u \leq 1\}$, and the base neural net is simply $\phi(x) = wx$ for some

scalar weight $w \in \mathbb{R}$ (i.e., w is the only neural net parameter in this case). Then if during neural net training, w becomes too large in absolute value (namely, $|w| > \frac{1}{\min_{i \neq j} \|X_i - X_j\|_2}$), then for all $i \neq j$,

$$\begin{aligned}
\mathbb{K}(X_i, X_j; \phi) &= K(\rho(X_i, X_j; \phi)) \\
&= \mathbb{1}\{\|\phi(X_i) - \phi(X_j)\|_2 \leq 1\} \\
&= \mathbb{1}\{\|wX_i - wX_j\|_2 \leq 1\} \\
&= \mathbb{1}\{|w|\|X_i - X_j\|_2 \leq 1\} \\
&= 0.
\end{aligned}$$

When this happens, the numerator and denominator of the leave-one-out training hazard estimator (2.6) are both 0, so by the earlier stated convention that $0/0 = 0$, the overall predicted hazard is 0 for all time. This will mean that the loss does not depend on the neural net parameter w at all, so the gradient of the loss with respect to w is 0. By simply using an infinite-support kernel function during training, we avoid having to deal with these zero kernel weight issues.

3. Scalable and Interpretable Test-Time Prediction with an Accuracy Guarantee

To make a prediction for test feature vector x, we would in principle have to compute the similarity between x and every training feature vector, which is computationally expensive for large training datasets. To address this problem, we apply *kernel netting* (Kpotufe and Verma, 2017) to deep kernel survival analysis, obtaining a model we call a *survival kernet*. Kernel netting constructs a compressed version of the training data for use at test time using the standard notion of ε-nets (e.g., see the textbook by Vershynin (2018)). As a technical remark, kernel netting was originally developed for classification and regression; extending the proof ideas to survival analysis requires carefully combining proof ideas by Kpotufe and Verma (2017) and Chen (2019). Separately, Kpotufe and Verma did not consider interpretability in their original kernel netting paper.

Sample splitting For our theoretical guarantee on test time prediction error later, we assume that the base neural net ϕ has already been trained on "pre-training" data $(X_1^\circ, Y_1^\circ, D_1^\circ)$, $\ldots$, $(X_{n_\circ}^\circ, Y_{n_\circ}^\circ, D_{n_\circ}^\circ)$ that are independent of training data $(X_1, Y_1, D_1), \ldots, (X_n, Y_n, D_n)$. As shorthand notation, we use $X_{1:n_\circ}^\circ := (X_1^\circ, \ldots, X_{n_\circ}^\circ) \in \mathcal{X}^{n_\circ}$, and similarly define $Y_{1:n_\circ}^\circ$, $D_{1:n_\circ}^\circ$, $X_{1:n}$, $Y_{1:n}$, and $D_{1:n}$. The pre-training data $(X_{1:n_\circ}^\circ, Y_{1:n_\circ}^\circ, D_{1:n_\circ}^\circ)$ need not be sampled in the same manner as the "proper" training data $(X_{1:n}, Y_{1:n}, D_{1:n})$. In practice, one could for example take a complete training dataset and randomly split it into two portions, the first portion to treat as the pre-training data, and the second portion to treat as the proper training data. After training ϕ on pre-training data, we refer to the learned neural net as $\widehat{\phi}$. Our theory treats how $\widehat{\phi}$ is learned as a black box, but requires that the output space of $\widehat{\phi}$ (the embedding space $\widetilde{\mathcal{X}}$) satisfy some regularity conditions. Later in our experiments, we also intentionally try setting the pre-training and training datasets to be the same although our theory does not cover this scenario.

Survival kernets We now state how to train and make predictions with a survival kernet. At test time, for a test feature vector x, we only consider using training data within a threshold distance τ from x, where distances are computed via the learned distance $\rho(x, x'; \widehat{\phi}) = \|\widehat{\phi}(x) - \widehat{\phi}(x')\|_2$ with pre-trained neural net $\widehat{\phi}$. In particular, at test time, we replace the function $\mathbb{K}$ from equation (2.5) with the "truncated" version

$$\widehat{\mathbb{K}}(x, x'; \widehat{\phi}) = K(\rho(x, x'; \widehat{\phi}))\mathbb{1}\{\rho(x, x'; \widehat{\phi}) \leq \tau\}. \tag{3.1}$$

From a computational viewpoint, we can take advantage of recent advances in (approximate) nearest neighbor search data structures for Euclidean distance to find neighbors of x that are within distance τ (e.g., Andoni et al. 2015; Andoni and Razenshteyn 2015; Malkov and Yashunin 2020; Prokhorenkova and Shekhovtsov 2020).

Training. The training procedure for a survival kernet works as follows, for a user-specified (approximate) Euclidean distance nearest neighbor data structure:

1. Learn base neural net ϕ with pre-training data $(X^\circ_{1:n_\circ}, Y^\circ_{1:n_\circ}, D^\circ_{1:n_\circ})$ by minimizing the deep kernel survival analysis loss L_{DKSA} given in equation (2.7) with minibatch gradient descent, *without truncating the kernel function.* Denote the learned neural net as $\widehat{\phi}$.
2. For the training (and not pre-training) feature vectors $X_{1:n}$, compute the embedding vectors $\widetilde{X}_1 = \widehat{\phi}(X_1), \widetilde{X}_2 = \widehat{\phi}(X_2), \ldots, \widetilde{X}_n = \widehat{\phi}(X_n)$, and construct a Euclidean-distance-based nearest neighbor data structure using training embedding vectors $\widetilde{X}_{1:n} := (\widetilde{X}_1, \widetilde{X}_2, \ldots, \widetilde{X}_n)$.
3. With the help of the nearest neighbor data structure, compute a subsample of $\widetilde{X}_{1:n}$ that we denote as $\widetilde{\mathcal{Q}}_\varepsilon \subseteq \widetilde{X}_{1:n}$, where $\varepsilon > 0$ is an approximation parameter (as $\varepsilon \to 0$, no subsampling is done, i.e., $\widetilde{\mathcal{Q}}_\varepsilon$ becomes $\widetilde{X}_{1:n}$). Specifically, $\widetilde{\mathcal{Q}}_\varepsilon$ is an ε-net; $\widetilde{\mathcal{Q}}_\varepsilon$ can be computed efficiently with the help of a nearest neighbor data structure as follows:
   (a) Initialize $\widetilde{\mathcal{Q}}_\varepsilon$ to be the empty set.
   (b) For $i \in \{1, 2, \ldots, n\}$: if $\widetilde{X}_i$'s nearest neighbor in $\widetilde{\mathcal{Q}}_\varepsilon$ is not within Euclidean distance ε, add $\widetilde{X}_i$ to $\widetilde{\mathcal{Q}}_\varepsilon$.
4. For each training embedding vector $\widetilde{X}_i$, assign it to a single closest exemplar point in $\widetilde{\mathcal{Q}}_\varepsilon$ (break ties arbitrarily). After this assignment, each exemplar point $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$ is assigned to a subset $\mathcal{I}_{\widetilde{q}} \subseteq \{1, 2, \ldots, n\}$ of training points. This could be viewed as a clustering assignment, where the training points of each cluster is represented by an exemplar.
5. For each exemplar point $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$, recalling that $t_1 < \cdots < t_m$ are the unique times of death in the training data, compute the following summary functions for $\ell = 1, \ldots, m$:

$$\mathbf{D}_{\widetilde{q}}(\ell) := \sum_{j \in \mathcal{I}_{\widetilde{q}}} D_j \mathbb{1}\{Y_j = t_\ell\}, \qquad \mathbf{R}^+_{\widetilde{q}}(\ell) := \sum_{j \in \mathcal{I}_{\widetilde{q}}} \mathbb{1}\{Y_j \geq t_\ell\}. \tag{3.2}$$

   Note that $\mathbf{D}_{\widetilde{q}}(\ell)$ is the number of deaths at time t_ℓ across training points assigned to exemplar $\widetilde{q}$, and $\mathbf{R}^+_{\widetilde{q}}(\ell)$ is the number of these training points that are "at risk" (could possibly die) at time t_ℓ.

Prediction. After training a survival kernet, prediction works as follows:

For test feature vector x, first compute the embedding vector $\widetilde{x} = \widehat{\phi}(x)$. Then form the hazard estimate

$$\widetilde{h}_{\widetilde{\mathcal{Q}}_\varepsilon}(\ell|\widetilde{x}) := \frac{\sum_{\widetilde{q}\in\widetilde{\mathcal{Q}}_\varepsilon} \widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q})\mathbf{D}_{\widetilde{q}}(\ell)}{\sum_{\widetilde{q}\in\widetilde{\mathcal{Q}}_\varepsilon} \widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q})\mathbf{R}^+_{\widetilde{q}}(\ell)} \qquad \text{for } \ell = 1, 2, \ldots, m, \tag{3.3}$$

where $\widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q}) := K(\|\widetilde{x}-\widetilde{q}\|_2)\mathbb{1}\{\|\widetilde{x}-\widetilde{q}\|_2 \le \tau\}$; as a reminder, $K : [0,\infty) \to [0,\infty)$ is a nonincreasing function (e.g., $K(u) = \exp(-u^2)$). Note that the nearest neighbor data structure constructed in step 2 of the training procedure can be used to find all exemplars in $\widetilde{\mathcal{Q}}_\varepsilon$ within distance τ of $\widetilde{x}$ (this is needed to compute $\widetilde{\mathbb{K}}$). The conditional survival function can be estimated by computing

$$\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) := \prod_{\ell=1}^{m}(1-\widetilde{h}_{\widetilde{\mathcal{Q}}_\varepsilon}(\ell|\widetilde{x}))^{\mathbb{1}\{t_\ell\le t\}} \qquad \text{for } t \ge 0. \tag{3.4}$$

As a corner case, if all the kernel weights are zero for test feature vector x, then we output the training set Kaplan-Meier survival function estimate (2.4) as the prediction.

The form of the predicted survival function and how it relates to the Kaplan-Meier estimator The predicted conditional survival function $\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}$ in equation (3.4) has the following interpretation. Consider an embedding vector $\widetilde{x} = \widehat{\phi}(x)$. Suppose that for some exemplar $\widetilde{q}' \in \widetilde{\mathcal{Q}}_\varepsilon$, the similarity score between $\widetilde{x}$ and $\widetilde{q}'$ is equal to $\widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q}') = 1$, whereas the similarity score between $\widetilde{x}$ and all other exemplars is 0. Put another way, the embedding vector $\widetilde{x}$ is "purely explained" by exemplar $\widetilde{q}'$ and none of the other exemplars. Then in this case, equation (3.3) would become

$$\widetilde{h}_{\widetilde{\mathcal{Q}}_\varepsilon}(\ell|\widetilde{x}) = \frac{\mathbf{D}_{\widetilde{q}'}(\ell)}{\mathbf{R}^+_{\widetilde{q}'}(\ell)} \qquad \text{for } \ell = 1, 2, \ldots, m,$$

and equation (3.4) would become

$$\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) = \prod_{\ell=1}^{m}(1-\widetilde{h}_{\widetilde{\mathcal{Q}}_\varepsilon}(\ell|\widetilde{x}))^{\mathbb{1}\{t_\ell\le t\}} = \prod_{\ell=1}^{m}\Big(1-\frac{\mathbf{D}_{\widetilde{q}'}(\ell)}{\mathbf{R}^+_{\widetilde{q}'}(\ell)}\Big)^{\mathbb{1}\{t_\ell\le t\}} \qquad \text{for } t \ge 0, \tag{3.5}$$

which is precisely the Kaplan-Meier estimator (equation (2.4)) restricted to training points that have been assigned to the cluster of exemplar $\widetilde{q}'$ (in Step 4 of the survival kernet training procedure).

In general, a data point x with embedding vector $\widetilde{x}$ can have a similarity score that is nonzero for multiple exemplars. This is akin to how in topic modeling, a data point could have nonzero weights for multiple topics. In topic modeling, a key visualization strategy is to focus on a single topic at a time and look at the distribution of features (i.e., words) that show up for that topic. This visualization strategy could be thought of as reasoning about what a data point would look like if it were to be purely explained by a single topic. In a similar vein, for our survival analysis setting, our visualization strategy later is based on reasoning about what a data point would look like if it were to be purely explained by a single cluster or exemplar (again, in our setting, a cluster directly corresponds to an exemplar).

Equation (3.5) reveals that a data point that is purely explained by exemplar $\widetilde{q}'$ would have a predicted conditional survival function that is just the Kaplan-Meier estimator restricted to data points in the cluster of $\widetilde{q}'$. Note that plotting such an estimated survival function is standard in survival analysis and the resulting plots are called *Kaplan-Meier curves*. Our visualization strategy later plots the Kaplan-Meier curve specific to each cluster/exemplar.

Connections to deep kernel survival analysis, to the original kernel netting, and to the original Kaplan-Meier estimator For a survival kernet model, note that if $\varepsilon = 0$ (i.e., the ε-net consists of all training embedding vectors), and the pre-training and training sets are set to be identical, then the model becomes an approximate version of the original deep kernel survival analysis model by Chen (2020). In particular, the approximation comes from the kernel function being truncated (set to be 0 beyond the critical threshold distance τ) at test time.

Separately, consider if the base neural net is the identity function $\phi(x) = x$ (which would require no pre-training data to learn, so we skip steps 1 and 2 of the survival kernet training procedure). Then in this case, we obtain a non-neural-net extension of the original kernel netting procedure by Kpotufe and Verma (2017) to the survival analysis setting.

Lastly, if $\varepsilon \to \infty$, then the ε-net $\widetilde{\mathcal{Q}}_\varepsilon$ would consist of a single exemplar that summarizes the whole training (and not pre-training) data. Put another way, there would only be a single cluster that consists of all the training data. If $\tau \to \infty$, then the prediction for any test point would simply be the original Kaplan-Meier survival function (Kaplan and Meier, 1958) fitted to the training data. The prediction would not actually depend on the base neural net ϕ in this case.

Outline for the remainder of this section In the remainder of this section, we discuss three main topics. First, we discuss model interpretability in Section 3.1. The key idea here has to do with the interpretation we described above of how if a data point were to be purely explained by a single exemplar $\widetilde{q}'$, then its predicted conditional survival function is just the Kaplan-Meier estimator restricted to training points in the cluster of $\widetilde{q}'$.

Next, we provide an overview of the theory we have developed for survival kernets in Section 3.2, starting from assumptions and the generalization error used to stating the main finite-sample guarantee and some interpretations and implications. In a nutshell, our theory says that if the embedding vectors are in some sense "nice" (satisfying standard theoretical assumptions made in nonparametric estimation and survival analysis), then a survival kernet estimates the conditional survival function arbitrarily accurately (up to some pre-specified time horizon) with high probability as the amount of training data grows large. The error bound we obtain has an order of growth that, ignoring a log factor, is optimal.

Lastly, we present a variant of our training procedure in Section 3.3 that turns out to yield significant accuracy improvements in practice although it is not covered by our theoretical analysis. This variant adds a step at the end of survival kernet training that fine-tunes the summary functions from training step 5. Specifically, note that the summary functions $\mathbf{D}_{\widetilde{q}}$ and $\mathbf{R}^+_{\widetilde{q}}$ in equation (3.2) are constructed from training data, and it could be that these are noisy or inaccurate. After training the base neural net ϕ and treating it as fixed, we could then set up a fine-tuning step in which we learn $\mathbf{D}_{\widetilde{q}}$ and $\mathbf{R}^+_{\widetilde{q}}$ in a neural net framework.

3.1 Model Interpretability

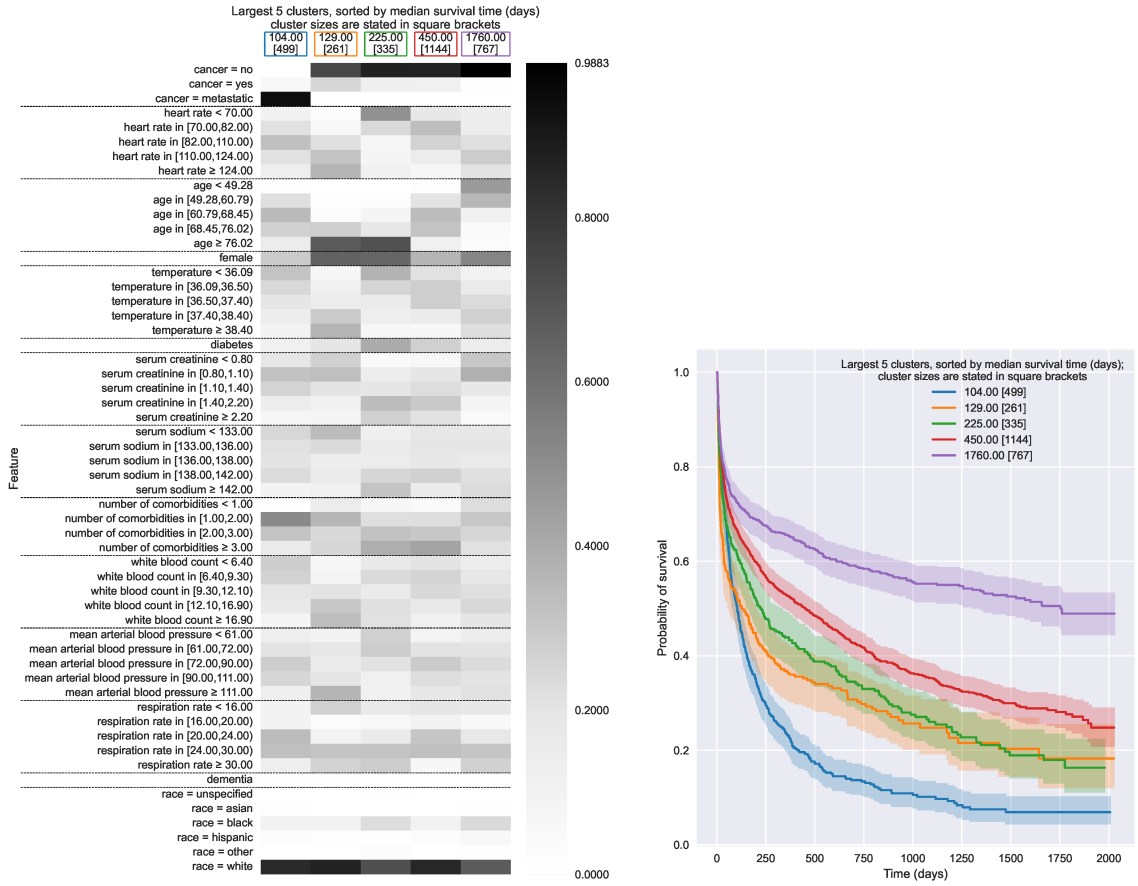
Visualizing clusters Recall that each exemplar $\widetilde{q}' \in \widetilde{\mathcal{Q}}_\varepsilon$ corresponds to a cluster of training points, i.e., there is a one-to-one correspondence between exemplars and clusters. For tabular data, we can create a heatmap visualization to help us quickly identify how clusters differ in terms of whether specific feature values are more prominent for specific clusters. In particular, we set the rows of the heatmap to correspond to different features, the columns to correspond to different clusters, and the heatmap intensity values to correspond to the fraction of points in a cluster with a specific feature value. As a concrete example, we show this heatmap visualization in Figure 1(a) for a dataset on predicting time until death of hospitalized patients from the Study to Understand Prognoses, Preferences, Outcomes, and Risks of Treatment (SUPPORT) (Knaus et al., 1995). For instance, we see that the leftmost cluster (column) in the heatmap corresponds to patients who often have metastatic cancer, who are mostly between 49.28 and 76.02 years of age, and who have at least one comorbidity. In contrast, the rightmost cluster corresponds to young patients without cancer. Our proposed heatmap visualization is a more visual way of conveying information like that of Chapfuwa et al. (2020) in their Table 2.

Moreover, per exemplar $\widetilde{q}' \in \widetilde{\mathcal{Q}}_\varepsilon$, we can compute its corresponding cluster's Kaplan-Meier survival curve using equation (3.5). As we pointed out previously, this predicted survival curve is precisely what the survival kernet model would predict for a data point that is purely explained by $\widetilde{q}'$ and none of the other exemplars. We can plot the survival curves of different exemplars as shown in Figure 1(b), where we have also included 95% confidence intervals using the standard exponential Greenwood formula (Kalbfleisch and Prentice, 1980); the x-axis is the time since a patient entered the study.

Note that the time (x-axis value) at which a Kaplan-Meier survival curve crosses probability 1/2 (y-axis value) corresponds to a median survival time estimate (since half the patients survived up to this time and the rest survive beyond this time). Thus, per cluster, we can obtain a median survival time estimate (if the cluster's Kaplan-Meier curve never crosses probability 1/2, then the median survival time is greater than the largest observed time for the cluster). In fact, for the heatmap in Figure 1(a), we have sorted the clusters (i.e., the columns) by median survival time from smallest to largest.

Comparing Kaplan-Meier curves across different groups of individuals is standard practice in survival analysis and gives a quick way to see which group is better or worse off over time. For instance, in this case, the purple cluster (with median survival time 1760 days) generally has higher survival probability across time compared to the four other clusters that have been plotted. Meanwhile, initially the orange cluster (median survival time 129 days) appears worse off than the green cluster (median survival time 225 days) but then these two clusters' Kaplan-Meier curves start heavily overlapping after around 1200 days since patients entered the study.

We remark that the SUPPORT dataset has patients within nine disease groups, of which three directly are related to cancer (colon cancer, lung cancer, multiple organ system failure with cancer). For these three disease groups, age and comorbidites are known to be predictors of survival (e.g., van Eeghen et al. 2015; Asmis et al. 2008; Frey et al. 2007). In this sense, the heatmap appears to surface some trends that agree with existing clinical literature. There are of course other variables present too. Overall, our proposed heatmap



(a) (b)

Figure 1: Visualization of the largest 5 clusters found by a survival kernet model trained on the SUPPORT dataset (we limit the number of clusters shown for ease of exposition and to prevent the plots from being too cluttered); more information on how the model is trained is in Section 5. Panel (a) shows a heatmap visualization that readily provides information on how the clusters are different, highlighting feature values that are prominent for specific clusters; the dotted horizontal lines separate features that correspond to the same underlying variable. Panel (b) shows Kaplan-Meier survival curves with 95% confidence intervals for the same clusters as in panel (a); the x-axis measures the number of days since a patient entered the study.

visualization and accompanying survival curve plot is meant as a debugging tool, to help a modeler see how the clusters relate to raw features and also to survival time distributions, revealing possible associations that may warrant additional investigation and that could be discussed with domain experts.

We provide a few technical details regarding how we generated Figure 1(a). Only for visualization purposes, continuous features have been discretized into 5 equal-sized bins (fewer bins are used if there are not enough data points in a cluster, or if the 20/40/60/80 percentile threshold values for a continuous feature are not all unique). The survival kernet

models themselves do *not* require continuous features to be discretized first. Moreover, we have sorted the features (i.e., rows of the heatmap) in the following manner: per row, we compute the intensity range (i.e., maximum minus minimum intensity values), and then we sort the rows from largest to smallest intensity range, with the constraint that rows corresponding to the same underlying variable (the same continuous feature that has been discretized, or the same categorical variable) are still grouped together. For example, the reason why the variable "cancer" shows up first in Figure 1(a) is that among the three different cancer rows, one of them has the highest intensity range across all heatmap rows.

Data-point-specific information Next, we observe that for any test feature vector x with embedding vector $\widetilde{x} = \widehat{\phi}(x)$, its hazard estimate is given by equation (3.3), reproduced below for ease of exposition:

$$\widetilde{h}_{\widetilde{\mathcal{Q}}_\varepsilon}(\ell|\widetilde{x}) = \frac{\sum_{\widetilde{q}\in\widetilde{\mathcal{Q}}_\varepsilon} \widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q})\mathbf{D}_{\widetilde{q}}(\ell)}{\sum_{\widetilde{q}\in\widetilde{\mathcal{Q}}_\varepsilon} \widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q})\mathbf{R}^+_{\widetilde{q}}(\ell)}.$$

In the numerator and denominator summations, the only exemplars $\widetilde{q}$ that contribute to the calculation are ones for which $\widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q})$ is positive. Again, since there is a one-to-one correspondence between exemplars and clusters, this means that each test feature vector's hazard is modeled by only a subset of the clusters.

For any test feature vector x, we can readily determine which exemplars/clusters could possibly contribute to the prediction for x by figuring out which $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$ satisfy $\widetilde{\mathbb{K}}(\widetilde{x},\widetilde{q}) > 0$. Of course, how large these weights are can also give a sense of the relative importance of the clusters for x. We could, for instance, make the same plots as in Figure 1 but only show the clusters that have nonzero weight for a specific test feature vector x.

3.2 Theory of Survival Kernets

We now provide an overview of our theoretical analysis of survival kernets. We begin with assumptions on the embedding space that are standard in nonparametric estimation theory (Section 3.2.1), although these assumptions are typically imposed on the raw feature space rather than the embedding space. We then state our assumption on how the embedding space relates to survival and censoring times (Section 3.2.2). All proofs are in Appendix A.

3.2.1 The Embedding Space

We assume that the raw feature vectors are sampled i.i.d. from a marginal distribution $\mathbb{P}_{\mathsf{X}}$. As we treat the pre-trained neural net $\widehat{\phi}$ as fixed, then the random embedding vector $\widetilde{X} = \widehat{\phi}(X)$ is sampled from some distribution $\mathbb{P}_{\widetilde{\mathsf{X}}}$ instead. We require $\mathbb{P}_{\widetilde{\mathsf{X}}}$ to satisfy some mild regularity conditions.

As an example, we remark that an embedding space that is uniform over a unit hypersphere (i.e., embedding vectors are Euclidean vectors with norm 1) satisfies all the assumptions of this subsection. For ways to encourage the learned embedding space to be uniform over the unit hypersphere, see the papers by Wang and Isola (2020) and Liu et al. (2021).

Our theory requires a technical assumption that ensures that the event that $\widetilde{X}$ lands within distance τ of an embedding vector $\widetilde{x} \in \widetilde{\mathcal{X}}$ has a well-defined probability. This event

could be phrased as $\widetilde{X}$ landing in the closed ball of radius τ centered at $\widetilde{x}$, denoted as $B(\widetilde{x}, \tau) := \{\widetilde{x}' \in \widetilde{\mathcal{X}} : \|\widetilde{x} - \widetilde{x}'\|_2 \leq \tau\}$.

Assumption A^{technical}**.** *Distribution $\mathbb{P}_{\widetilde{\mathsf{X}}}$ is a Borel probability measure with compact support $\widetilde{\mathcal{X}} \subseteq \mathbb{R}^d$.*

Compactness of $\widetilde{\mathcal{X}}$ eases the exposition. Our results trivially extend to the case where embedding vectors sampled from $\mathbb{P}_{\widetilde{\mathsf{X}}}$ land in a compact region (where our theory applies) with probability at least $1 - \delta$ for some $\delta \geq 0$, and otherwise when the embedding vectors land outside the compact region, we tolerate a worst-case prediction error with probability δ.

We use the standard notion of covers, packings, and nets (all specialized to Euclidean distance).

Definition 1 *For any radius $\varepsilon > 0$:*

- *A subset $\widetilde{\mathcal{Q}}$ of $\widetilde{\mathcal{X}}$ is called an* ε-cover *of $\widetilde{\mathcal{X}}$ if for any $\widetilde{x} \in \widetilde{\mathcal{X}}$, there exists $\widetilde{q} \in \widetilde{\mathcal{Q}}$ such that $\|\widetilde{x} - \widetilde{q}\|_2 \leq \varepsilon$.*
- *A subset $\widetilde{\mathcal{Q}}$ of $\widetilde{\mathcal{X}}$ is called an* ε-packing *of $\widetilde{\mathcal{X}}$ if for any two distinct $\widetilde{q}$ and $\widetilde{q}'$ in $\widetilde{\mathcal{Q}}$, we have $\|\widetilde{q} - \widetilde{q}'\|_2 > \varepsilon$.*
- *A subset $\widetilde{\mathcal{Q}}$ of $\widetilde{\mathcal{X}}$ is called an* ε-net *of $\widetilde{\mathcal{X}}$ if $\widetilde{\mathcal{Q}}$ is both an ε-cover and an ε-packing of $\widetilde{\mathcal{X}}$.*

Next, our theory makes use of the standard complexity notions of covering numbers and intrinsic dimension to describe the embedding space, where lower complexity will correspond to tighter error bounds. As an illustrative example, we explain how these behave when the embedding space is the unit hypersphere $\mathbb{S}^{d-1} = \{\widetilde{x} \in \mathbb{R}^d : \|\widetilde{x}\|_2 = 1\}$ (with $d \geq 2$).

Definition 2 *The* ε-covering number *of $\widetilde{\mathcal{X}}$ (for Euclidean distance) is the smallest size possible for an ε-cover for $\widetilde{\mathcal{X}}$. We denote this number by $N(\varepsilon; \widetilde{\mathcal{X}})$.*

The embedding space $\widetilde{\mathcal{X}}$ being compact implies that for every $\varepsilon > 0$, we have $N(\varepsilon; \widetilde{\mathcal{X}}) < \infty$, and moreover, that $\widetilde{\mathcal{X}}$ has a finite diameter $\Delta_{\widetilde{\mathcal{X}}} := \max_{\widetilde{x}, \widetilde{x}' \in \widetilde{\mathcal{X}}} \|\widetilde{x} - \widetilde{x}'\|_2 < \infty$. For example, the unit hypersphere $\mathbb{S}^{d-1}$ clearly has diameter 2. Moreover, its covering number is bounded as follows.

Claim 3 (Follows from Corollary 4.2.13 by Vershynin (2018)) *For any $\varepsilon > 0$, the unit hypersphere has covering number $N(\varepsilon; \mathbb{S}^{d-1}) \leq (\frac{4}{\varepsilon} + 1)^d$.*

Covering numbers provide a way to derive error bounds. Another way is to assume a known "intrinsic dimension" of the embedding space. We use the following notion of intrinsic dimension.

Assumption A^{intrinsic}**.** *There exist a positive integer $d' > 0$ (called the* intrinsic dimension *of $\mathbb{P}_{\widetilde{\mathsf{X}}}$) and positive constants $C_{d'}$ and r^* such that for any $\widetilde{x} \in \widetilde{\mathcal{X}}$ and $r \in (0, r^*]$, we have $\mathbb{P}_{\widetilde{\mathsf{X}}}(B(\widetilde{x}, r)) \geq C_{d'} r^{d'}$.*

When embedding vectors are on the unit hypersphere, then under a mild regularity condition, the embedding space has intrinsic dimension $d - 1$.

Claim 4 *If $\mathbb{P}_{\widetilde{\mathsf{X}}}$ has a probability density function that is 0 outside of $\mathbb{S}^{d-1}$ and lower-bounded by a constant $c_{\min} > 0$ over $\mathbb{S}^{d-1}$, then the embedding space has intrinsic dimension $d - 1$.*

We remark that for embedding vectors on the hypersphere, one desirable property is that they are uniformly distributed (Wang and Isola, 2020; Liu et al., 2021), which is of course a special case of Claim 4.

3.2.2 Relating Embedding Vectors to Survival and Censoring Times

Next, we impose a survival analysis assumption by Chen (2019), which is a slight variant on an earlier assumption by Dabrowska (1989).

Assumption $\mathbf{A}^{survival}$. *In addition to Assumption $\mathbf{A}^{technical}$, we further assume that the conditional survival, censoring, and observed time distributions $\mathbb{P}_{\mathsf{T}|\widetilde{\mathsf{X}}}$, $\mathbb{P}_{\mathsf{C}|\widetilde{\mathsf{X}}}$, and $\mathbb{P}_{\mathsf{Y}|\widetilde{\mathsf{X}}}$ correspond to continuous random variables with PDF's $\widetilde{f}_{\mathsf{T}|\widetilde{\mathsf{X}}}$, $\widetilde{f}_{\mathsf{C}|\widetilde{\mathsf{X}}}$, and $\widetilde{f}_{\mathsf{Y}|\widetilde{\mathsf{X}}}$. The conditional survival function that we aim to predict is precisely $\widetilde{S}(t|\widetilde{x}) := \int_t^\infty \widetilde{f}_{\mathsf{T}|\widetilde{\mathsf{X}}}(s|\widetilde{x})ds$. To be able to estimate this function for any $\widetilde{x} \in \widetilde{\mathcal{X}}$, we assume that censoring does not almost surely happen for every $\widetilde{x} \in \widetilde{\mathcal{X}}$. Moreover, we assume the following:*

(a) (Observed time distribution has large enough tails) For a user-specified time $t_{horizon} > 0$, there exists a constant $\theta > 0$ such that $\int_{t_{horizon}}^\infty f_{\mathsf{Y}|\widetilde{\mathsf{X}}}(s|\widetilde{x})ds \geq \theta$ for all $\widetilde{x} \in \widetilde{\mathcal{X}}$.

(b) (Smoothness: embedding vectors that are close by should have similar survival time distributions and also similar censoring time distributions) PDF's $\widetilde{f}_{\mathsf{T}|\widetilde{\mathsf{X}}}$ and $\widetilde{f}_{\mathsf{C}|\widetilde{\mathsf{X}}}$ are assumed to be Hölder continuous with respect to embedding vectors, i.e., there exist constants $\lambda_{\mathsf{T}} > 0$, $\lambda_{\mathsf{C}} > 0$, and $\alpha > 0$ such that for all $\widetilde{x}, \widetilde{x}' \in \widetilde{\mathcal{X}}$,

$$|\widetilde{f}_{\mathsf{T}|\widetilde{\mathsf{X}}}(t|\widetilde{x}) - \widetilde{f}_{\mathsf{T}|\widetilde{\mathsf{X}}}(t|\widetilde{x}')| \leq \lambda_{\mathsf{T}}\|\widetilde{x} - \widetilde{x}'\|_2^\alpha,$$
$$|\widetilde{f}_{\mathsf{C}|\widetilde{\mathsf{X}}}(t|\widetilde{x}) - \widetilde{f}_{\mathsf{C}|\widetilde{\mathsf{X}}}(t|\widetilde{x}')| \leq \lambda_{\mathsf{C}}\|\widetilde{x} - \widetilde{x}'\|_2^\alpha.$$

For any estimate $\widetilde{G}$ of the true conditional survival function $\widetilde{S}$, our theory uses the generalization error

$$\mathcal{L}_{\text{survival}}(\widetilde{G}, \widetilde{S}) := \mathbb{E}_{\widetilde{X}\sim\mathbb{P}_{\widetilde{\mathsf{X}}}}\Big[\frac{\int_0^{t_{\text{horizon}}} \big(\widetilde{G}(t|\widetilde{X}) - \widetilde{S}(t|\widetilde{X})\big)^2 dt}{t_{\text{horizon}}}\Big].$$

We only aim to accurately estimate $\widetilde{S}(t|\widetilde{x})$ up to the user-specified time t_{horizon} (that appears in Assumption $\mathbf{A}^{\text{survival}}$(a)). We shall plug in the survival kernet conditional survival function estimate $\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}$ (from step 6 of the survival kernet procedure) in place of $\widetilde{G}$.

3.2.3 Theoretical Guarantee on Prediction Accuracy

We are now ready to state the main theoretical result of the paper, which states a generalization error bound for survival kernets.

Theorem 5 *Suppose that Assumptions $\mathbf{A}^{technical}$, $\mathbf{A}^{intrinsic}$, and $\mathbf{A}^{survival}$ hold, and we train a survival kernet with $\varepsilon = \beta\tau$ when constructing $\widetilde{\mathcal{Q}}_\varepsilon$, where $\beta \in (0,1)$ and $\tau > 0$ are user-specified parameters, and the truncated kernel used is of the form in equation (3.1). Let*

$\Psi = \min\{N(\frac{(1-\beta)\tau}{2}; \widetilde{\mathcal{X}}), \frac{1}{C_{d'}((1-\beta)\tau)^{d'}}\}$, *where* d' *is the intrinsic dimension. Then when* $n \geq \mathcal{O}(\frac{K(0)}{K(\tau)((1-\beta)\tau)^{d'}})$,

$$\mathbb{E}_{\widetilde{X}_{1:n}, Y_{1:n}, D_{1:n}}[\mathcal{L}_{survival}(\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}, \widetilde{S})] \leq \widetilde{\mathcal{O}}\Big(\frac{1}{n} \cdot \frac{K^4(0)}{K^4(\tau)} \cdot \Psi\Big) + (1+\beta)^{2\alpha}\tau^{2\alpha} \cdot \mathcal{O}\Big(\frac{K^2(0)}{K^2(\tau)}\Big),$$

where $\widetilde{\mathcal{O}}$ *ignores log factors.*

The error bound consists of two terms, which correspond to variance and bias respectively. When the embedding space has intrinsic dimension d', this error bound is, up to a log factor, optimal in the case where $K(u) = 1$ for $u \geq 0$, and $\tau = \widetilde{\mathcal{O}}(n^{-1/(2\alpha+d')})$. In this scenario, the survival error bound is $\widetilde{\mathcal{O}}(n^{-2\alpha/(2\alpha+d')})$, which matches the lower bound for conditional CDF estimation by Chagny and Roche (2014) (this is a special case of the survival analysis setup in which there is no censoring). Of course, in practice, K is typically chosen to decay, so τ should be chosen to not be too large due to the error bound's dependence on the ratio $K(0)/K(\tau)$.

High-level proof overview We combine proof ideas of Kpotufe and Verma (2017) and Chen (2019). What these authors consider to be the raw feature space is what we take to be the embedding space (the output space of the already trained neural net $\widehat{\phi}$). In this high-level proof overview, we only highlight some of the key ideas of our proof.

The bulk of the proof treats the test point x as fixed. Its corresponding embedding vector is $\widetilde{x} = \widehat{\phi}(x)$. Recall that $\widetilde{\mathcal{Q}}_\varepsilon$ is the set of exemplars in the embedding space, and that step 4 of the survival kernet training procedure assigns each exemplar point $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$ to a subset $\mathcal{I}_{\widetilde{q}} \subset \{1, 2, \ldots, n\}$ of training points. In particular, every training point is assigned to exactly one exemplar so that

$$\underbrace{\{1, 2, \ldots, n\}}_{\text{training data indices}} = \coprod_{\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon} \mathcal{I}_{\widetilde{q}},$$

where "$\coprod$" denotes a disjoint union. Following Kpotufe and Verma (2017), an initial key observation is that the only training points that can possibly contribute to the prediction for $\widetilde{x}$ (i.e., they have nonzero truncated kernel weight) are the ones with indices in

$$\mathcal{N}_{\widetilde{\mathcal{Q}}_\varepsilon}(\widetilde{x}) := \coprod_{\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon \text{ s.t. } \|\widetilde{x} - \widetilde{q}\|_2 \leq \tau} \mathcal{I}_{\widetilde{q}}.$$

Kpotufe and Verma's analysis focuses on regression in which they then proceed to analyzing a kernel-weighted average of the labels of the training points in $\mathcal{N}_{\widetilde{\mathcal{Q}}_\varepsilon}(\widetilde{x})$. In the survival analysis setting, the challenge is that prediction is more complicated than taking a kernel-weighted average of Y_i's.

To address predicting the conditional survival function $\widetilde{S}(\cdot|\widetilde{x})$, the proof strategy by Chen (2019) uses a Taylor expansion to decompose the predicted log of the estimated conditional survival function $\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(\cdot|\widetilde{x})$ into three terms:

$$\log \widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) = W_1(t|\widetilde{x}) + W_2(t|\widetilde{x}) + W_3(t|\widetilde{x}),$$

where $W_1(t|\widetilde{x})$ is a kernel-weighted average of a hypothetical label that we do not directly observe (we provide more details on this shortly), $W_2(t|\widetilde{x})$ is an error term for how well we can approximate the CDF of the distribution of observed times specific to test embedding vector $\widetilde{x}$, and lastly $W_3(t|\widetilde{x})$ is a higher-order Taylor series error term. In more detail, $W_1(t|\widetilde{x})$ is a kernel-weighted average, where the i-th training point's label is taken to be $-\frac{D_i \mathbb{1}\{Y_i \leq t\}}{\widetilde{S}(Y_i|\widetilde{x})}$; note that this label knows the true conditional survival function $\widetilde{S}(\cdot|\widetilde{x})$. This hypothetical label only shows up in the proof and is not actually needed when implementing the model. Similarly, the CDF estimation problem that $W_2(t|\widetilde{x})$ relates to also only shows up in the proof and does not need to be implemented.

Chen (2019) does not use a ε-net and his analysis could be thought of as the case where we take $\varepsilon \to 0$ for the ε-net (so that every training embedding vector is its own exemplar). For this special case, Chen showed sufficient conditions that ensure that $W_1(t|\widetilde{x}) \to \log \widetilde{S}(t|\widetilde{x})$, $W_2(t|\widetilde{x}) \to 0$, and $W_3(t|\widetilde{x}) \to 0$. Thus, $\log \widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) \to \log \widetilde{S}(t|\widetilde{x})$, which can be turned into a statement on the difference $\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) - \widetilde{S}(t|\widetilde{x})$, without logs.

Extending Chen's analysis to account for ε-nets requires a bit of extra bookkeeping and, ultimately, the two main changes are as follows (that at a high-level relate to variance and bias of the survival kernet estimator):

1. Chen's original analysis (restated to use our notation) regularly involves the quantity $n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x}, \tau))$, which is the expected number of training points that land within distance τ of $\widetilde{x}$. Again, keep in mind that the setup Chen considers could be thought of as every training point being its own exemplar. Thus, the exemplars/training points that could contribute to the prediction of $\widetilde{x}$ is indeed any training point whose embedding vector lands within distance τ of $\widetilde{x}$. Thus, when the quantity $n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x}, \tau))$ is larger, then we should have more training points that contribute to the prediction of $\widetilde{x}$, thus reducing the variance in the estimated conditional survival function.

   However, once we use ε-nets with $\varepsilon = \beta\tau$, the complication is that now, the training points that could contribute to the prediction of $\widetilde{x}$ need to be assigned to exemplars that are within distance τ of $\widetilde{x}$. It is possible that a training point x' with embedding vector $\widetilde{x}'$ satisfies $\|\widetilde{x}' - \widetilde{x}\| \leq \tau$ but $\widetilde{x}'$ could be assigned to an exemplar that is farther away from distance τ from $\widetilde{x}$ so that training point x' does not actually contribute to the prediction for $\widetilde{x}$. Here, the key observation (which shows up in Kpotufe and Verma's regression analysis but is more generally a standard ε-net proof technique) is that it suffices to consider training points whose embedding vectors are within distance $(1-\beta)\tau$ of $\widetilde{x}$. These particular embedding vectors will for sure be assigned to exemplar(s) within distance τ of $\widetilde{x}$: any such embedding vector is within distance $(1-\beta)\tau$ from $\widetilde{x}$ and can be assigned to an exemplar that is at most an additional distance $\beta\tau$ away from $\widetilde{x}$ since the set of exemplars is an $(\beta\tau)$-net. Thus, whereas Chen's original analysis depended on the quantity $n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x}, \tau))$, now we instead replace this quantity with $n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x}, (1-\beta)\tau))$ as we want the expected number of training embedding vectors within distance $(1-\beta)\tau$ (which get assigned to exemplar(s) within distance τ of $\widetilde{x}$) to be large.

2. The other main change is that in Chen's original analysis, there is a bias calculation involving Hölder's inequality. Restated using our notation, the idea is that a training

embedding vector that is a distance τ away from $\widetilde{x}$ incurs a bias in estimating $W_1(t|\widetilde{x})$ that scales roughly as $\tau^{2\alpha}$ (specifically, see Lemma F.4 of Chen (2019); note that Chen uses sup norm error so the exponent is just α instead of 2α whereas in this paper, we use squared error which adds the factor of 2). However, now that we are using an ε-net, the prediction for $\widetilde{x}$ could depend on an exemplar that is a distance τ away from $\widetilde{x}$, and this exemplar could have a training embedding vector assigned to it that is a worst case distance of $\tau + \beta\tau = (1+\beta)\tau$ away from $\widetilde{x}$. Thus, the worst-case bias incurred is now $((1+\beta)\tau)^{2\alpha}$ instead of just $\tau^{2\alpha}$. This same idea shows up in Kpotufe and Verma's regression analysis as well.

Overall, we get that for large enough n, with high probability,

$$\mathbb{E}_{\substack{\text{training}\\ \text{data}}}\left[\frac{1}{t_{\text{horizon}}}\int_0^{t_{\text{horizon}}}(\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x}) - \widetilde{S}(t|\widetilde{x}))^2 dt\right] \le \widetilde{\mathcal{O}}\Big(\frac{1}{n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x},(1-\beta)\tau))} + ((1+\beta)\tau)^{2\alpha}\Big).$$

We then further take the expectation of both sides over the randomness in sampling the test embedding vector $\widetilde{x} \sim \mathbb{P}_{\widetilde{\mathsf{X}}}$. Doing so, the first term on the right-hand side $\mathbb{E}_{\widetilde{x}\sim\mathbb{P}_{\widetilde{\mathsf{X}}}}[\frac{1}{n\mathbb{P}_{\widetilde{\mathsf{X}}}(\mathcal{B}(\widetilde{x},(1-\beta)\tau))}]$ can be upper-bounded by $\frac{1}{n}\cdot\Psi$ that shows up in the statement of Theorem 5. For details, see the full proof in Appendix A.3.

3.3 Summary Fine-Tuning

Our theoretical analysis crucially relies on the summary functions being the ones stated in equation (3.2). However, in practice, it turns out that learning the summary functions improves prediction accuracy. We now discuss how to do this. Specifically, we add a "summary fine-tuning" step that refines the summary functions $\mathbf{D}_{\widetilde{q}}(\ell)$ and $\mathbf{R}^+_{\widetilde{q}}(\ell)$ that are used in the test-time hazard predictor (given in equation (3.3)).

To do summary fine-tuning, we treat everything in the survival kernet model as fixed except for the summary functions $\mathbf{D}_{\widetilde{q}}(\ell)$ and $\mathbf{R}^+_{\widetilde{q}}(\ell)$ for each exemplar $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$ and each time step $\ell \in \{1,\dots,m\}$, where as a reminder the time steps $t_1 < t_2 < \cdots < t_m$ are at the unique times of death in the training data. We now use the survival kernet predicted hazard function given in equation (3.3), which we plug into the deep kernel survival analysis loss L_{DKSA} given in equation (2.7). Importantly, now when we minimize the loss L_{DKSA}, we are not learning the base neural net ϕ from earlier (as it is treated as fixed); we are only learning the summary functions. Thus, the only additional details needed in explaining how summary fine-tuning works are in how we parameterize the summary functions. This requires a little bit of care to encode the constraints that the number of people at risk monotonically decreases over time and is always at least as large as the number of people who die over time.

Parameterizing summary functions for number of deaths We parameterize the summary function $\mathbf{D}_{\widetilde{q}}(\ell)$ for the number of deaths at time step ℓ specific to exemplar $\widetilde{q}$ as

$$\mathbf{D}_{\widetilde{q}}(\ell) := \exp(\gamma_{\widetilde{q},\ell}) + \exp(\gamma_\ell^{\text{baseline}}),$$

where $\gamma_{\widetilde{q},\ell} \in \mathbb{R}$ and $\gamma_\ell^{\text{baseline}} \in \mathbb{R}$ are unconstrained neural net parameters. We initialize $\mathbf{D}_{\widetilde{q}}(\ell)$ to be approximately equal to the value given in equation (3.2) by using the initial

values

$$\gamma_{\widetilde{q},\ell} = \log\Big(\sum_{j \in \mathcal{I}_{\widetilde{q}}} D_j \mathbb{1}\{Y_j = t_\ell\}\Big), \tag{3.6}$$

$$\gamma_\ell^{\text{baseline}} = \log(10^{-12}) \approx -27.631. \tag{3.7}$$

Note that the 10^{-12} number in equation (3.7) is just an arbitrarily chosen small constant. Meanwhile, in computing equation (3.6), to prevent numerical issues, if the summation inside the log is less than 10^{-12}, then we also set $\gamma_{\widetilde{q},\ell} = \log(10^{-12})$.

Parameterizing summary functions for number of individuals at risk To parameterize the summary function for the number of individuals at risk, we first introduce a new variable for the number of individuals censored at time step ℓ specific to exemplar $\widetilde{q}$:

$$\mathbf{C}_{\widetilde{q}}(\ell) := \exp(\omega_{\widetilde{q},\ell}) + \exp(\omega_\ell^{\text{baseline}}),$$

where $\omega_{\widetilde{q},\ell} \in \mathbb{R}$ and $\omega_\ell^{\text{baseline}} \in \mathbb{R}$ are unconstrained neural net parameters. How these parameters are initialized works the same way as for the number of deaths and requires that we count how many people are censored at each time step ℓ, which is straightforward to compute from the training data (the only minor complication is that the time steps are for unique observed times of death, and times of censoring could happen at other times—the simple fix is to consider the number of individuals censored at time step ℓ to be summed across all individuals censored at times within the interval $(t_{\ell-1}, t_\ell]$).

Finally, it suffices to note that the number of individuals at risk at time step ℓ specific to exemplar $\widetilde{q}$ is given by the recurrence relation

$$\mathbf{R}_{\widetilde{q}}^+(\ell) = \mathbf{D}_{\widetilde{q}}(\ell) + \mathbf{C}_{\widetilde{q}}(\ell) + \mathbf{R}_{\widetilde{q}}^+(\ell+1), \tag{3.8}$$

where $\mathbf{R}_{\widetilde{q}}^+(m+1) := 0$. In particular, for a specific exemplar $\widetilde{q}$, once we have computed $\mathbf{D}_{\widetilde{q}}(\ell)$ and $\mathbf{C}_{\widetilde{q}}(\ell)$ for all ℓ, then we can easily compute $\mathbf{R}_{\widetilde{q}}(m)$, $\mathbf{R}_{\widetilde{q}}(m-1)$, $\mathbf{R}_{\widetilde{q}}(m-2)$, …, $\mathbf{R}_{\widetilde{q}}(1)$ using equation (3.8)—note that we compute these in reverse chronological order.

Final remarks on summary fine-tuning To summarize, summary fine-tuning computes refined estimates of the summary functions $\mathbf{D}_{\widetilde{q}}(\ell)$ and $\mathbf{R}_{\widetilde{q}}^+(\ell)$ by minimizing the deep kernel survival analysis loss L_{DKSA} (equation (2.7)) using the hazard function given in equation (3.3). We treat the pre-trained base neural net $\widehat{\phi}$ and survival kernet clustering assignments as fixed. In fact, the only neural net parameters that we optimize over are the newly introduced variables $\gamma_{\widetilde{q},\ell}, \gamma_\ell^{\text{baseline}}, \omega_{\widetilde{q},\ell}, \omega_\ell^{\text{baseline}} \in \mathbb{R}$ for $\widetilde{q} \in \widetilde{\mathcal{Q}}_\varepsilon$ and $\ell \in \{1, 2, \ldots, m\}$.

In terms of model interpretation, summary fine-tuning does not change survival kernet cluster assignments and only changes the summary functions $\mathbf{D}_{\widetilde{q}}(\ell)$ and $\mathbf{R}_{\widetilde{q}}^+(\ell)$ of the different clusters. For a specific exemplar $\widetilde{q}$, we could still compute a survival curve for the cluster corresponding to $\widetilde{q}$ using equation (3.5), where we plug in the refined summary functions (instead of the original summary functions stated in equation (3.2)). The survival curve per cluster could still be interpreted as what the survival kernet model would predict for a data point that is purely explained by a single cluster and none of the other clusters. However, by using the refined summary functions, equation (3.5) no longer corresponds to a Kaplan-Meier curve, so we can no longer use the 95% confidence intervals that are meant for Kaplan-Meier survival curves.

We remark that simultaneously optimizing the summary functions *and* the base neural net ϕ would be difficult in terms of how we set up the survival kernet framework because we only determine the exemplars after the base neural net ϕ has been learned and treated as fixed (the exemplars are chosen based on the embedding space). If we update the base neural net, the exemplars would change as would their summary functions, and in fact even the number of exemplars (and thus, the number of summary functions) could change. An alternating optimization could potentially be done (i.e., alternate between optimizing ϕ, choosing which training points are the exemplars, and then optimizing the summary functions) but would be computationally costly; for simplicity, we do not do such an optimization.

4. Scalable Tree Ensemble Warm-Start

Whereas Section 3 focused on constructing a test-time predictor that can scale to large datasets, we now turn to accelerating the training of the base neural net ϕ, which happens in the very first step of survival kernet training. Specifically, we now present a warm-start strategy for initializing ϕ prior to optimizing the parameters of ϕ by minimizing the DKSA loss L_{DKSA} given in equation (2.7).

Our warm-start strategy begins by learning a kernel function using a scalable tree ensemble approach such as XGBOOST (Chen and Guestrin, 2016). Because a kernel function learned by a tree ensemble is not represented as a neural net, we then fit a neural net to the tree ensemble's kernel function via minibatch gradient descent. In short, we warm-start ϕ using a *Tree ensemble Under a Neural Approximation* (TUNA). After the warm-start, we then fine-tune ϕ using the DKSA loss L_{DKSA} (this is not to be confused with summary fine-tuning; here we in fact are fine-tuning the base neural net ϕ and we are *not* optimizing over any sort of summary functions). Importantly, at the end of this section, we explain how TUNA can accelerate neural architecture search. Note that our theory in Section 3.2 trivially supports TUNA: simply train base neural net ϕ with TUNA on the pre-training data.

Approximating a tree ensemble kernel The key idea behind TUNA is that decision trees and their ensembles implicitly learn kernel functions (Breiman, 2000). Thus, by using any scalable decision tree or decision tree ensemble learning approach, we can learn an initial kernel function guess $\mathbb{K}_0$. For example, a decision tree has $\mathbb{K}_0(x, x') = \mathbb{1}\{x \text{ and } x' \text{ are in the same leaf}\}$. For a decision forest $\mathbb{K}_0(x, x')$ is equal to the fraction of trees for which x and x' are in the same leaf. For gradient tree boosting, the kernel function is a bit more involved and the general case is given by Chen and Shah (2018, Section 7.1.3). For XGBOOST when we add one tree at a time and the final ensemble has equal weight across trees, then it suffices to set $\mathbb{K}_0(x, x')$ to be the fraction of trees for which x and x' are in the same leaf.

The kernel function $\mathbb{K}_0$ of a tree ensemble does not give us an embedding representation of the data, where the embedding function is a neural net. However, we can train the base neural net ϕ to approximate $\mathbb{K}_0$ by minimizing the mean-squared error

$$\frac{1}{n_\circ(n_\circ - 1)/2} \sum_{i=1}^{n_\circ} \sum_{j=i+1}^{n_\circ} \Big(\underbrace{\mathbb{K}(X_i^\circ, X_j^\circ)}_{\text{to be learned}} - \underbrace{\mathbb{K}_0(X_i^\circ, X_j^\circ)}_{\substack{\text{given by the} \\ \text{tree ensemble}}} \Big)^2. \tag{4.1}$$

As a reminder, "$\circ$" indicates that a quantity is part of pre-training data, and $\mathbb{K}$ depends on the base neural net ϕ: $\mathbb{K}(x, x') = K(\|\phi(x) - \phi(x')\|_2^2)$, where for example $K(u) = \exp(-u^2)$ for a Gaussian kernel. To scale to large datasets, we minimize loss (4.1) in minibatches. The TUNA warm-start strategy is as follows:

1. Train a scalable tree ensemble (e.g., XGBOOST) on the pre-training data; denote its learned kernel function by $\mathbb{K}_0$. For a subset $\mathcal{S} \subseteq \{X_1^\circ, \ldots, X_{n_\circ}^\circ\}$ of the pre-training feature vectors, let $\mathbf{K}_\mathcal{S}$ denote the $|\mathcal{S}|$-by-$|\mathcal{S}|$ Gram matrix formed so that the (i, j)-th entry is given by $\mathbb{K}_0(x_i^\circ, x_j^\circ)$ where x_i° and x_j° are the i-th and j-th pre-training feature vectors in $\mathcal{S}$ (with elements of $\mathcal{S}$ ordered arbitrarily).
2. For each minibatch consisting of pre-training feature vectors $x_1^\circ, \ldots, x_b^\circ$ where b is the batch size:

(a) Compute the batch's tree ensemble Gram matrix $\mathbf{K}_{\{x_1^\circ, \ldots, x_b^\circ\}}$ (defined in step 1) using $\mathbb{K}_0$.

(b) Compute the current neural net's Gram matrix estimate $\widehat{\mathbf{K}}_{\{x_1^\circ, \ldots, x_b^\circ\}}$, which has (i, j)-th entry given by $\mathbb{K}(x_i^\circ, x_j^\circ) = K(\|\phi(x_i^\circ) - \phi(x_j^\circ)\|_2^2)$.

(c) Let this minibatch's loss be the MSE loss (4.1) restricted to feature vectors of the current minibatch, i.e., the MSE loss between $\widehat{\mathbf{K}}_{\{x_1^\circ, \ldots, x_b^\circ\}}$ and $\mathbf{K}_{\{x_1^\circ, \ldots, x_b^\circ\}}$. Update parameters of neural net ϕ based on the gradient of this minibatch's loss.

As a slight refinement of this warm-start procedure, in our experiments later, between steps 1 and 2, we initialize the base neural net ϕ using a different warm-start strategy by Chen (2020) that is based on multidimensional scaling (MDS) (Borg and Groenen, 2005) (i.e, within our warm-start strategy, we further use another warm-start strategy). Chen's warm-start strategy is computationally expensive to compute, so we use this strategy only over a small subset of the training data. For details, see Appendix B.

Neural architecture search Step 2 of TUNA can be run using different base neural nets (and step 1 only needs to be run once). Thus, we can search over different base neural net architectures and choose whichever one achieves the lowest average minibatch loss (the one described in step 2(c) above). Then when fine-tuning by minimizing the DKSA loss L_{DKSA}, we do not repeat a search over neural net architectures. Put another way, when searching over neural net architectures (whether using grid search or a more sophisticated approach to selecting architectures to try), we focus on minimizing a batched version of the loss (4.1). We remark that step 2 of TUNA does a simple least squares fit without accounting for any survival analysis problem structure.

5. Experiments

Our experiments are designed to help us understand the empirical performance of survival kernets in terms of prediction accuracy (Section 5.1) and computation time (Section 5.2). We consider survival kernets with and without the summary fine tuning, and also with and without our TUNA warm-start procedure. We also provide additional examples of cluster visualizations for different datasets that we use (Section 5.3). Our code is publicly available.[2]

2. `https://github.com/georgehc/survival-kernets`

Table 5.1: Dataset characteristics.

Dataset	Number of data points	Number of features	Censoring rate
ROTTERDAM/GBSG	2,232	7	43.2%
SUPPORT	8,873	14 (19*)	31.97%
UNOS	62,644	49 (127*)	50.23%
KKBOX	2,814,735	15 (45*)	34.67%

* number of features after preprocessing (note that ROTTERDAM/GBSG dataset's preprocessing does not change the number of features)

Datasets We benchmark survival kernets on four standard survival analysis datasets. We provide basic characteristics of these datasets in Table 5.1, where we have sorted the datasets in increasing number of data points. The first three datasets are on predicting time until death. The first dataset ROTTERDAM/GBSG is for breast cancer patients and technically actually consists of two separate datasets: the Rotterdam tumor bank dataset (Foekens et al., 2000) and the German Breast Cancer Study Group dataset (Schumacher et al., 1994). However, these datasets share a common set of features and are frequently analyzed together (e.g., Katzman et al. 2018; Kvamme et al. 2019; Chen 2020; Zhong et al. 2021), so for the rest of the paper, we denote these as the single dataset ROTTERDAM/GBSG. The next dataset SUPPORT is the one we already mentioned in Section 3.1 that is for hospitalized patients from the Study to Understand Prognoses, Preferences, Outcomes, and Risks of Treatment (SUPPORT) (Knaus et al., 1995). The third dataset UNOS is for patients receiving heart transplants from the United Network for Organ Sharing.[3] The fourth dataset KKBOX is the one on customer churn mentioned in Section 1. These datasets have appeared in literature although not necessarily all at once in the same paper (e.g., Chapfuwa et al. 2018; Giunchiglia et al. 2018; Katzman et al. 2018; Lee et al. 2018; Kvamme et al. 2019; Chen 2020; Nagpal et al. 2021; Zhong et al. 2021). Data preprocessing details are in Appendix C.1. For the ROTTERDAM/GBSG dataset, following Katzman et al. (2018), we treat the Rotterdam data as the training data and the German Breast Cancer Study Group data as the test data. For the other datasets, we use a random 70%/30% train/test split.

Baseline survival models As baselines, we use an elastic-net-regularized Cox model (denoted as ELASTIC-NET COX), XGBOOST (Chen and Guestrin, 2016), DEEPSURV (Katzman et al., 2018), DEEPHIT (Lee et al., 2018), Deep Cox mixtures (abbreviated as DCM) (Nagpal et al., 2021), and DKSA (Chen, 2020). All neural net models use a multilayer perceptron as the base neural net. Hyperparameter grids and optimization details are in Appendix C.2. Note that our hyperparameter grid for ELASTIC-NET COX includes no regularization, which corresponds to the standard Cox model (Cox, 1972). We remark that DEEPHIT has been previously reported to obtain state-of-the-art accuracy on the largest three datasets we have tested (cf., Lee et al. 2018; Kvamme et al. 2019; as these authors use different experimental setups, their accuracy numbers are not directly comparable to each other or to ours; however, our accuracy results yield trends similar to what was found by these authors in terms of how the baseline methods compare to each other).

3. We use the UNOS Standard Transplant and Analysis Research data from the Organ Procurement and Transplantation Network as of September 2019, requested at: `https://www.unos.org/data/`

Variants of survival kernets For survival kernets, as part of hyperparameter tuning, we try a number of variants. We always use the Gaussian kernel $K(u) = \exp(-u^2)$. We fix the truncation distance to be $\tau = \sqrt{2 \log 10} \approx 2.146$ (i.e., training points contributing to prediction for a test point must have kernel weight/similarity score with the test point that is at least $\exp(-\tau^2) = \exp(-(\sqrt{2 \log 10})^2) = 0.01$), and we further only consider at most 128 approximate nearest neighbors, which are found using the HNSW algorithm (Malkov and Yashunin, 2020). We comment on different choices of τ in Appendix C.5. For constructing ε-nets, we set $\varepsilon = \beta\tau$ and try $\beta \in \{1/2, 1/4\}$. For the embedding space, motivated by Claims 3 and 4, we constrain the embedding vectors to be on a hypersphere (in preliminary experiments, we found that leaving the embedding space unconstrained would occasionally lead to numerical issues during training and did not yield better validation accuracy scores than having a hypersphere constraint). We try standard neural net initalization (He et al., 2015) with an exhaustive hyperparameter grid/neural architecture sweep vs our strategy TUNA (denoted in tables as KERNET vs TUNA-KERNET).

When dividing data so that pre-training and training data are not the same, we split the full training data into 60%/20%/20% pre-training/training/validation sets (neural net training uses the pre-training and validation sets, with the latter used for early stopping). When we use this sample splitting, we add the suffix "(SPLIT)" to the method name in tables (e.g., "TUNA-KERNET (SPLIT)" refers to a survival kernet model trained with TUNA warm-start and with pre-training and training sets disjoint). Otherwise, if the pre-training and training sets are set to be the same, then we split the full training data into 80%/20% training/validation sets similar to all the baselines, using the validation set for hyperparameter tuning (including early stopping during neural net training); in this case, we add the suffix "(NO SPLIT)" to the method name in tables.

Finally, if summary fine-tuning (abbreviated SFT) is used, then we also add the suffix "SFT". For instance, "(NO SPLIT, SFT)" means that pre-training and training data are set to be the same, and summary fine-tuning is used.

Experimental setup For every dataset and for every model, we repeat the following basic experiment 5 times (except for a select few cases where the model is too computationally expensive to run):

1. We randomly split the dataset's training set into an 80% "proper" training set and a 20% validation set.

2. We train the model on the proper training set using different hyperparameters. The validation set is used to select the best hyperparameter according to an accuracy metric (we specifically use the time-dependent concordance index C^{td} by Antolini et al. (2005)).

3. For the model achieving the best validation set accuracy, we use it to predict on the current dataset's test set. We record the test set accuracy score achieved (again using C^{td} index).

In our results later, we report the mean and standard deviation of the test set accuracy scores per dataset and per method across the 5 repetitions of the above basic experiment (we indicate the few cases where a model is too computationally expensive to run so that either we only ran the model once or the model could not even finish running a single time).

Table 5.2: Test set C^{td} indices (mean ± standard deviation across 5 experimental repeats, except for a few cases that run into computational issues).

Model	Dataset			
	ROTTERDAM/GBSG	SUPPORT	UNOS	KKBOX
ELASTIC-NET COX	0.6660 ± 0.0045	0.6046 ± 0.0013	0.5931 ± 0.0011	0.8438 ± 0.0001
XGBOOST	0.6703 ± 0.0128	0.6281 ± 0.0031	0.6028 ± 0.0009	0.8714 ± 0.0000
DEEPSURV	**0.6850** ± 0.0160	0.6155 ± 0.0032	0.5941 ± 0.0021	0.8692 ± 0.0003
DEEPHIT	0.6792 ± 0.0121	0.6354 ± 0.0047	0.6170 ± 0.0016	**0.9148** ± 0.0001
DCM	0.6763 ± 0.0104	0.6289 ± 0.0047	0.6101 ± 0.0023	0.8830*
DKSA	0.6570 ± 0.0139	0.6316 ± 0.0080	out of memory	out of memory
KERNET (SPLIT)	0.6450 ± 0.0086	0.5934 ± 0.0073	0.5936 ± 0.0039	0.8933*
KERNET (SPLIT, SFT)	0.6478 ± 0.0140	0.6007 ± 0.0040	0.5984 ± 0.0039	0.9027*
KERNET (NO SPLIT)	0.6599 ± 0.0190	0.6244 ± 0.0026	0.6033 ± 0.0039	0.8942*
KERNET (NO SPLIT, SFT)	0.6621 ± 0.0191	0.6291 ± 0.0059	0.6071 ± 0.0039	0.9029*
TUNA-KERNET (SPLIT)	0.6510 ± 0.0212	0.6220 ± 0.0026	0.6028 ± 0.0032	0.8952 ± 0.0002
TUNA-KERNET (SPLIT, SFT)	0.6544 ± 0.0239	0.6287 ± 0.0050	0.6105 ± 0.0046	0.9049 ± 0.0004
TUNA-KERNET (NO SPLIT)	0.6694 ± 0.0163	0.6385 ± 0.0038	0.6130 ± 0.0029	0.8957 ± 0.0005
TUNA-KERNET (NO SPLIT, SFT)	0.6719 ± 0.0135	**0.6426** ± 0.0045	**0.6211** ± 0.0025	0.9057 ± 0.0003

* we only ran one experimental repeat due to how computationally expensive DCM and the non-TUNA survival kernet variants are, so standard deviations are unavailable for these

As a technical detail, for the first step above, we set random seeds so that per dataset, the 5 experimental repeats' have different random splits for the proper training and validation sets. However, we do *not* use different random splits across the different models. For example, for the very first experimental repeat on ROTTERDAM/GBSG, every model would use the exact same proper training set and also the exact same validation set. We set up our experiments in this manner so that every model gets access to the same random proper training/validation splits across the 5 experimental repeats.

5.1 Results on Prediction Accuracy

We report test set C^{td} indices in Table 5.2 (mean ± standard deviation across 5 experimental repeats unless some computational issue was encountered, as described above). The main findings are as follows:

(a) On the largest three datasets (SUPPORT, UNOS, and KKBOX), the TUNA-KERNET (NO SPLIT, SFT) model (i.e., a survival kernet trained with TUNA, pre-training data set to be the same as training data, and summary fine-tuning) achieves the highest mean test set C^{td} indices on SUPPORT and UNOS and the second highest on KKBOX (second to DEEPHIT).

(b) On the smallest dataset ROTTERDAM/GBSG, TUNA-KERNET (NO SPLIT, SFT) has higher accuracy than the other survival kernet variants but the mean C^{td} score it achieves is lower than those of several baselines (DEEPSURV, DEEPHIT, DCM—of which DEEPSURV achieves the highest mean C^{td} score). TUNA-KERNET (NO SPLIT, SFT) does not appear significantly less accurate than these baselines though since the test set C^{td} standard deviations for ROTTERDAM/GBSG across all models are high. For

example, for the test set C^{td} indices of TUNA-KERNET (NO SPLIT, SFT), the interval that is within one standard deviation of the mean (namely, 0.6719 ± 0.0135) contains the mean test set C^{td} indices achieved by DEEPSURV, DEEPHIT, and DCM.[4]

(c) Our experimental results include two variants of survival kernets that do correspond to our theoretical analysis (namely KERNET (SPLIT) and TUNA-KERNET (SPLIT), of which the latter generally outperforms the former). We point out that TUNA-KERNET (SPLIT) does outperform a few baselines: it achieves a higher mean test set C^{td} index than ELASTIC-NET COX and DEEPSURV on SUPPORT, UNOS, and KKBOX. For the KKBOX dataset, TUNA-KERNET (SPLIT) also outperforms XGBOOST and DCM.

(d) Focusing only on the survival kernet variants, we observe the following general trends:

i. By making the pre-training and training sets disjoint as required by our theory, the accuracy decreases (this trend generally holds for any variant of survival kernets where the only difference is whether pre-training and training sets are disjoint vs set to be the same).

ii. Using summary fine-tuning improves accuracy compared to setting the summary functions based on equation (3.2) (this trend generally holds for any variant of survival kernets where the only difference is whether summary fine-tuning is used).

iii. Using the TUNA warm-start strategy improves accuracy compared to standard neural net initialization (this trend generally holds for any variant of survival kernets where the only difference is whether TUNA is used).

As a minor remark, our KKBOX C^{td} index is significantly higher for DEEPHIT than previously reported in literature (e.g., Kvamme et al. 2019) because we also try setting the time discretization grid to be all unique times of customer churn in the dataset, which turns out to significantly improve DEEPHIT's accuracy for the KKBOX dataset (when we discretize to 64 or 128 time steps, the C^{td} indices we get are on par with what Kvamme et al. get).

5.2 Results on Computation Time

We next report wall clock computation times for training survival kernet variants and the baselines in Table 5.3. These training times are inclusive of hyperparameter tuning (which includes searching over neural net architectures). Our computation time results aim to show how much our TUNA warm-start strategy accelerates training and also to give a sense of the time needed to train the different models under consideration. All code was run under similar conditions on identical Ubuntu 20.04.2 LTS instances, each with an Intel Core i9-10900K CPU (3.70 GHz, 10 cores, 20 threads), 64GB RAM, and an Nvidia Quadro RTX 4000 (8GB GPU RAM). These compute instances accessed code and data that were centrally stored on a single network-attached storage system.

4. In fact, when we look at the 5 experimental repeats (as we described in the experimental setup above, for a specific experimental repeat, all models use the same proper training set and the same validation set), DEEPSURV achieved higher test set C^{td} index compared to TUNA-KERNET (NO SPLIT, SFT) only in 3 out of the 5 experiments: specifically, DEEPSURV achieved test set C^{td} indices **0.693**, **0.697**, **0.699**, 0.673, and 0.663 whereas TUNA-KERNET (NO SPLIT, SFT) achieved 0.673, 0.649, 0.674, **0.679**, and **0.685**, respectively.

Table 5.3: Total training time (mean ± standard deviation across 5 experimental repeats, except for a few cases that run into computational issues), which includes training using different hyperparameters. Note that TUNA-KERNET (SPLIT) and TUNA-KERNET (SPLIT, SFT) include the time needed to train an XGBOOST warm-start model only using *pre-training data*, whereas TUNA-KERNET (NO SPLIT) and TUNA-KERNET (NO SPLIT, SFT) include time to train XGBOOST on the *full training data*.

Model	Total training time (minutes)			
	ROTTERDAM/GBSG	SUPPORT	UNOS	KKBOX
ELASTIC-NET COX	0.294 ± 0.047	0.762 ± 0.031	3.075 ± 0.169	60.171 ± 0.155
XGBOOST	9.369 ± 0.209	17.885 ± 0.537	73.503 ± 0.420	292.358 ± 1.943
DEEPSURV	0.140 ± 0.009	0.323 ± 0.016	2.784 ± 0.209	48.621 ± 0.136
DEEPHIT	5.083 ± 0.343	9.984 ± 0.373	467.479 ± 5.621	1573.683 ± 5.333
DCM	6.739 ± 0.539	20.667 ± 0.829	1111.717 ± 25.459	38263.568*
DKSA	2.756 ± 0.184	31.135 ± 0.269	out of memory	out of memory
KERNET (SPLIT)	8.917 ± 0.464	47.581 ± 1.354	841.923 ± 80.509	10323.367*
KERNET (SPLIT, SFT)	9.074 ± 0.480	49.025 ± 1.149	851.257 ± 86.945	10515.057*
KERNET (NO SPLIT)	23.964 ± 0.588	128.579 ± 3.241	1579.229 ± 58.450	14323.791*
KERNET (NO SPLIT, SFT)	24.103 ± 0.583	129.694 ± 3.094	1595.272 ± 65.490	14673.548*
TUNA-KERNET (SPLIT)	7.239 ± 0.289	18.510 ± 1.339	160.845 ± 10.788	1586.327 ± 26.143
TUNA-KERNET (SPLIT, SFT)	7.500 ± 0.366	19.921 ± 1.758	169.935 ± 7.993	1929.540 ± 16.990
TUNA-KERNET (NO SPLIT)	12.762 ± 0.733	32.220 ± 1.109	320.571 ± 33.236	2088.604 ± 86.066
TUNA-KERNET (NO SPLIT, SFT)	12.941 ± 0.708	33.770 ± 1.801	328.397 ± 31.003	2560.418 ± 65.785

* we only ran one experimental repeat due to how computationally expensive DCM and the non-TUNA survival kernet variants are, so standard deviations are unavailable for these

Note that the timing results in Table 5.3 are fair in comparing between the different variants of survival kernets but *not* fair in comparing between a survival kernet variant and a baseline or between two different baselines. The reason for this is that, as aforementioned, we report training times that are inclusive of hyperparameter tuning. Different models can have drastically different hyperparameters and hyperparameter grid sizes. We can easily increase or decrease the number of hyperparameters we try for a model, which can significantly increase or decrease the overall training time of the model as a result. Because we use the exact same hyperparameter grid across survival kernet variants, comparing computation times across them is fair. However, comparing the training times of a survival kernet variant and of a baseline (or even of two different baselines) does not lead to a fair comparison due to the hyperparameter grids being different.[5] That said, we present the overall training times across models anyways in Table 5.3 (mean ± standard deviation across the 5 experimental repeats unless some computational issue was encountered).

5. Note that reporting "per epoch" training times does not give a fair comparison either since, for instance, training the TUNA-KERNET (NO SPLIT, SFT) model involves three different kinds of neural net minibatch gradient descent optimizations (the first is for approximating the XGBOOST tree ensemble kernel, the second is for fine-tuning the base neural net using the DKSA loss, and the third is for summary fine-tuning) whereas, for instance, the baselines DEEPSURV and DEEPHIT each only involve one kind of neural net minibatch gradient descent optimization. Reporting "per hyperparameter setting" training times is not straightforward since, for instance, the TUNA warm-start strategy could be viewed as further expanding the hyperparameter grid to include the hyperparameters of the base tree ensemble model being trained while at the same time never doing an exhaustive grid search.

The main takeaways from Table 5.3 are as follows:

(a) For each TUNA-KERNET variant, when we compare it to its corresponding variant that does not use TUNA, then we consistently find that using TUNA dramatically reduces computation time, with savings of 17.3% to 85.4% depending on the specific variant of survival kernets used and on the dataset. The savings are larger on the larger datasets; for instance, when only considering the largest three datasets, then the savings actually range from 59.4% to 85.4%.

(b) Using summary fine-tuning clearly increases training time compared to not using it, with an increase of 0.58% to 22.6% additional time depending on the specific variant of survival kernets used and on the dataset.

(c) Even though survival kernets with TUNA are significantly faster to train than survival kernets without TUNA, survival kernets with TUNA are still relatively expensive to train when compared to baselines tested (aside from DKSA and DCM, both of which appear to have issues scaling to larger datasets) at least for the hyperparameter grids that we have used.[6]

We reiterate that there are trivial ways to reduce or increase the computation time of either survival kernets or any of the baseline models by simply removing or adding hyperparameter settings to try. For example, we suspect that for the survival kernet variants and for DEEPHIT, we could remove some of the hyperparameter settings that we had tried while retaining a similar test set accuracy. Note that the hyperparameter grid we used for every survival kernet variant is a superset of the one we used for DEEPHIT, so we expected the computation times to be higher for survival kernet variants than DEEPHIT. It turns out though that all the survival kernet variants with TUNA have lower overall training times than DEEPHIT on the UNOS dataset for the hyperparameter grids we used.

As an illustrative example to provide more insight on how much time is spent on different parts of survival kernet training, we provide a timing breakdown for the TUNA-KERNET (NO SPLIT, SFT) model trained on the KKBOX dataset in Table 5.4. Note that in this table, we have separated the computation times for using different ε values in ε-net construction (as a reminder, we set $\varepsilon = \beta\tau$ where the threshold distance is set to be $\tau = \sqrt{2\log(10)} \approx 2.146$, and we sweep over $\beta \in \{1/2, 1/4\}$). For example, we see that if we only tried using $\beta = 1/4$ and did not try $\beta = 1/2$, then we would reduce the computation time by about 31% in this case. In Table 5.4, we also provide some detail on the hyperparameters that we sweep over for the different steps during training. As a reminder, for the TUNA warm-start strategy, the neural net architecture is treated as fixed after we approximate the XGBOOST kernel with the base neural net (in particular, trying different neural net architectures happens during the step labeled "Approximate XGBOOST kernel with base neural net" in Table 5.4).

6. For DCM in particular, the maximum number of clusters we attempted to use was 6 (which was also the maximum used by the original authors (Nagpal et al., 2021)) and even so, the total training time clearly grows at a rate that makes it impractical for use on large datasets (as shown in Table 5.3, a single experiment on KKBOX took over 26 days to run). The issue is that the default behavior in the code by Nagpal et al. is that every 10 minibatches, it does posterior inference across the entire proper training set (not just the current minibatch) to update every cluster's survival curve (represented as a spline).

Table 5.4: Time breakdown in training the TUNA-KERNET (NO SPLIT, SFT) model on the KKBOX dataset for our experimental setup (mean ± standard deviation across 5 experimental repeats). The time breakdown for training TUNA-KERNET (NO SPLIT) is the same without the final summary fine-tuning time. Note that our TUNA warm-start strategy corresponds to the combination of the first two tasks listed below. Also, note that every task listed below involves tuning different hyperparameters. For details on hyperparameters, see Appendix C.2. For a slightly more detailed time breakdown that separates the fine-tuning of the base neural net further to include the different numbers of time steps to discretize to, see Appendix C.3.

Task	Hyperparameter settings	Time (minutes)
Train XGBOOST model	192*	292.358 ± 1.943
Approximate XGBOOST kernel with base neural net	18†	111.987 ± 0.307
Fine-tune base neural net with DKSA loss ($\beta = 1/2$)	36‡	793.884 ± 52.890
Fine-tune base neural net with DKSA loss ($\beta = 1/4$)	36‡	890.374 ± 34.188
Summary fine-tuning	2§	471.814 ± 37.123
Total		2560.418 ± 65.785

* corresponds to the full XGBOOST hyperparameter grid that we use
† base neural net hyperparameters (number of hidden layers, nodes per hidden layer), learning rate
‡ survival loss hyperparameters (η and σ_{rank} from equation (2.7)), number of time steps to discretize to, learning rate
§ learning rate

5.3 Cluster Visualizations

We now present cluster visualizations of TUNA-KERNET (NO SPLIT, SFT) models. For simplicity, we only show visualizations for the first experimental repeat.[7] For ease of exposition, we begin with the SUPPORT dataset since we had already presented a figure for it in Section 3.1 (Figure 1). We now also discuss some modified versions of the two plots in Figure 1, first to how incorporate summary fine-tuning and then, separately, to discuss how to summarize *all* the clusters found and not just a few. After providing cluster visualizations for the SUPPORT dataset, we then provide visualizations for the ROTTERDAM/GBSG, UNOS, and KKBOX datasets.

SUPPORT dataset For the final TUNA-KERNET (NO SPLIT) model trained on the SUPPORT dataset that we used to evaluate test set accuracy, we visualize the largest 5 clusters in Figure 1 using our visualization strategies from Section 3.1. In Section 3.1, we had already pointed out how one could interpret the different clusters, for instance finding the rightmost cluster in Figure 1(a) to be of mostly young, cancer-free patients. Meanwhile, the Kaplan-Meier curves in Figure 1(b) quickly give us a sense of which clusters appear to be better or worse off than others over time. For this dataset, the largest 5 clusters contain

7. As is standard in machine learning literature, when we talk about model interpretability, we are talking about interpreting a specific trained model. For example, consider topic models (such as latent Dirichlet allocation (Blei et al., 2003) or any neural topic model (Zhao et al., 2021)) or prototypical part networks (Chen et al., 2019). When one re-trains such a topic or prototypical part model using different random seeds (but leave model hyperparameters fixed), the topics/prototypes learned can vary across random seeds. Even so, each specific learned model (per experimental repeat) can be interpreted.

60.5% of the proper training data. There are a total of 73 clusters, many of which are small. Of course, more clusters can be visualized at once; the plots get more cluttered though.

If instead summary fine-tuning is used and we want to visualize the largest 5 clusters of the final TUNA-KERNET (NO SPLIT, SFT) model, then we would make the following changes. First, note that the final TUNA-KERNET (NO SPLIT, SFT) model is actually identical to the final TUNA-KERNET (NO SPLIT) model except that summary fine-tuning is used at the very end, which does not change the survival kernet cluster assignments from the ε-net. Thus, in terms of cluster visualization, the only main change would be that instead of computing the survival curve for each cluster using a Kaplan-Meier plot as in Figure 1(b), we would instead compute the survival curve using equation (3.5) with the learned summary functions. Next, based on these newly estimated survival curves, we can look at when they cross probability 1/2 to obtain median survival time estimates. The survival curves from summary fine-tuning in this case are shown in Figure 2; note that these curves are in the exact same ordering and correspond to the same clusters as in Figure 1 (the same color means the same cluster although the median survival times change), but as we can readily see, they look slightly different from the Kaplan-Meier curves from Figure 1(b). Since the survival curves in Figure 2 are no longer the standard Kaplan-Meier ones, we do not have confidence intervals for them. Meanwhile, the heatmap in Figure 1(a) remains the same except that the median survival times at the top would be updated to the new values computed (and that appear in the legend of Figure 2); note that the columns may have to be reordered if we still want them to be sorted by median survival time (the new median survival time estimates based on summary fine-tuning do not have to be in the same order as the original median survival time estimates based on Kaplan-Meier curves).

We have found that for some clusters, the survival curve from summary fine-tuning is relatively close to that of the Kaplan-Meier estimate without summary fine-tuning whereas for other clusters, there could be a larger difference (such as the green cluster shifting upward at earlier times in Figure 2 compared to in Figure 1(b)). Some possible explanations could be that the green cluster is in a region of the embedding space that is close to other clusters and its survival function that gets learned using summary fine-tuning could be thought of as being estimated from a larger region of the embedding space (rather than just getting averaged over points assigned to the green cluster by the ε-net), or that even among the points in the cluster, calculating the survival curve by weighting every point equally (as in the standard Kaplan-Meier survival curve estimate) does not make as much sense as using some sort of unequal/nonuniform weighting.

Although there are a total of 73 clusters, we have shown only 5 of them thus far. While we could visualize all 73 at once using the same visualization ideas as above, the heatmap and survival curve plots would get cluttered. One way to still visualize information from all 73 clusters is to use agglomerative clustering (Hastie et al., 2009, Section 14.3.12) to merge many clusters into larger "superclusters". For the new superclusters, we can make visualizations similar to Figure 1. For instance, using standard complete-linkage agglomerative clustering, we partition the 73 clusters into 10 superclusters in Figure 3. The resulting feature heatmap in Figure 3(a) gives a more comprehensive view of how features vary across the entire training dataset. In this case, the second-to-rightmost and third-to-rightmost columns/clusters in the heatmap each only have one data point, suggesting that they are outliers among the training data. The rest of the superclusters, however,

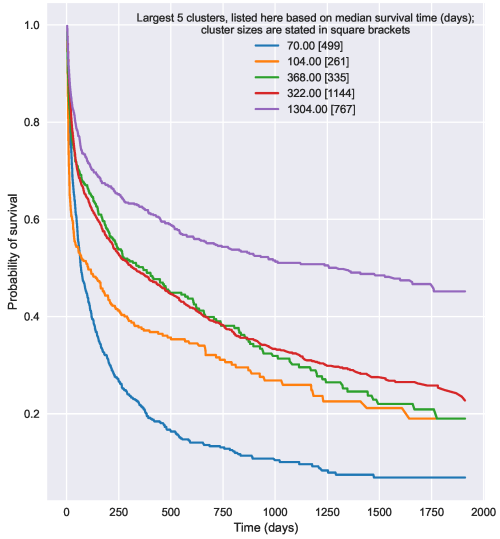
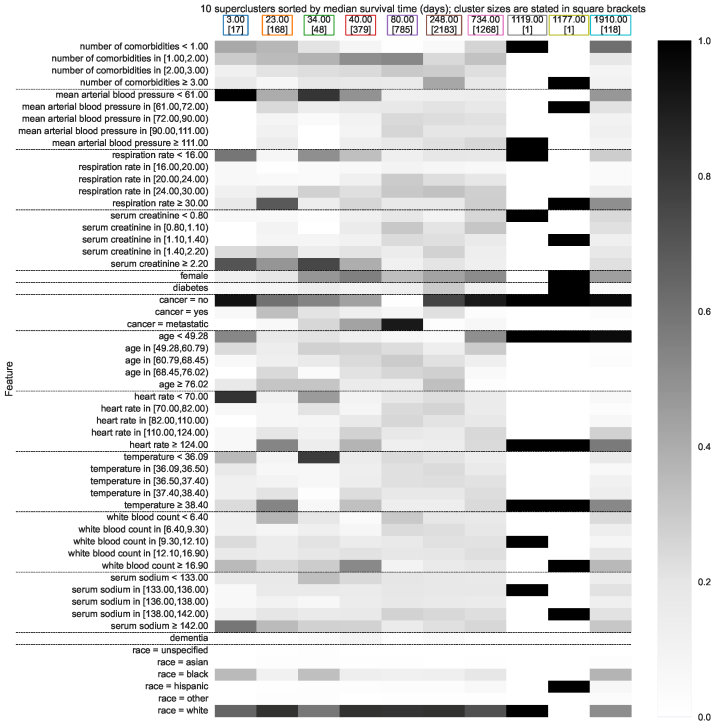


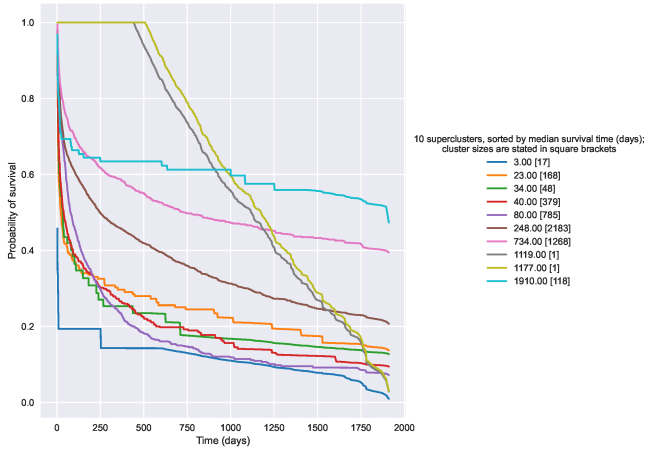
Figure 2: Survival curves for the largest 5 clusters found by the final TUNA-KERNET (NO SPLIT, SFT) model trained on the SUPPORT dataset; the x-axis measures the number of days since a patient entered the study. Note that the green curve has a higher median survival time estimate than the red curve; this is not a typo in that we are ordering the clusters the exact same way as in Figure 1(b). In particular, the median survival time estimates using summary fine-tuning do *not* have to be ordered the same way as the median survival time estimates from the Kaplan-Meier estimator.

exhibit some of the same trends we pointed out earlier when only looking at the largest 5 clusters, such as the rightmost supercluster (with the highest median survival time) tending to be for younger, cancer-free patients who often have no comorbidities. Among the leftmost superclusters, we get different common patterns in patient characteristics that are associated with low survival times (e.g., the leftmost four superclusters have unusually high creatinine levels possibly indicative of severe kidney disorder, the supercluster with a median survival time of 80 days corresponds to patients tending to have metastatic cancer and at least one comorbidity). Meanwhile, the survival curves in Figure 3(b) are more spread out compared to those in Figure 2.

Note that when we merge clusters into a supercluster, there is a question of how to compute an aggregated survival curve for the supercluster. If we are not using summary fine-tuning, then the aggregation is straightforward: we continue to use the standard Kaplan-Meier survival curve estimator but instead now use data from all patients in a supercluster to compute that supercluster's survival curve. However, when using summary fine-tuning, there is no standard way to aggregate information from different clusters' learned summary functions. For simplicity, to come up with the survival curves in Figure 3(b), for the clusters that are being merged into a supercluster, we take these clusters' survival curves and then just take a weighted average to obtain the survival curve for the supercluster (the weights are proportional to how many points are in each cluster belonging to the supercluster). The median survival times are then estimated by finding the times in which these survival curves cross probability 1/2.



(a)



(b)

Figure 3: Visualization of 10 superclusters for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the SUPPORT dataset. These 10 superclusters summarize all 73 clusters found by the TUNA-KERNET (NO SPLIT, SFT) model by merging clusters using complete-linkage agglomerative clustering. Panel (a) shows a feature heatmap visualization. Panel (b) shows survival curves for the same superclusters as in panel (a); the x-axis measures the number of days since a patient entered the study. The second-to-rightmost and third-to-rightmost columns/clusters each only have one data point, suggesting that they are outliers.

Rotterdam-GBSG dataset Next, we plot the feature heatmap and survival curves for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the ROTTERDAM/GBSG dataset (technically trained only on the Rotterdam portion of the data) in Figure 4. In this case the final model only has a total of 10 clusters, so we plot all 10 of them. This number of clusters found is also small enough that using the supercluster idea above that we discussed for the SUPPORT dataset is unnecessary.

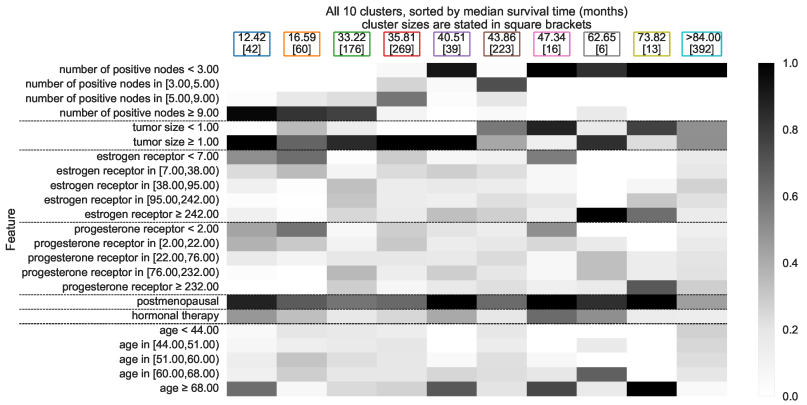
From looking at the heatmap in Figure 4(a), we immediately see a few key patterns: lower survival times are associated with higher numbers of positive nodes (these are lymph nodes with cancer) and higher tumor sizes. The rightmost cluster has the highest median survival time (> 64 months); note that the reason the median survival time is not a precise estimate here and is only a lower bound is that the estimated survival curve for this cluster never crosses the probability 1/2 threshold by the last observed time. For this rightmost cluster, we see that the number of positive nodes is low and also the age distribution of patients in the cluster also tends to be younger than those of the other clusters. These qualitative findings agree with previous literature on breast cancer (Sopik and Narod, 2018).

By looking instead at the survival curves in Figure 4(b), we immediately notice a pattern in the survival curves: they roughly look like powers of each other, which happens when the proportional hazards assumption holds! This observation agrees with the prediction accuracy results from earlier where we saw that the DEEPSURV model (which uses a proportional hazards assumption) is the best-performing model we evaluated for ROTTERDAM/GBSG. We suspect that the survival kernet model is essentially too flexible of model in this case (arguably DEEPHIT and DCM are also too flexible for this dataset).

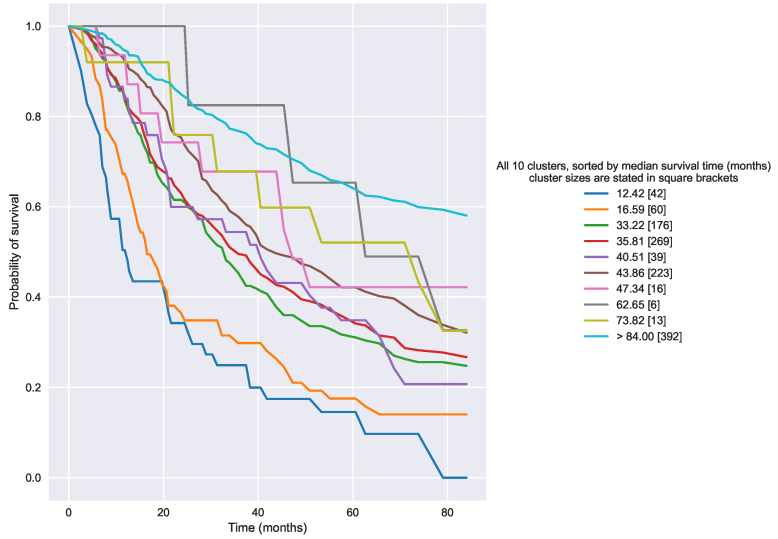
UNOS dataset For the UNOS dataset, automatic hyperparameter tuning led to a final TUNA-KERNET (NO SPLIT, SFT) model with a total of 30 clusters that altogether contain 35,080 training points. The largest 5 clusters only account for 87.9% of the proper training data. For these largest 5 clusters, we plot their feature heatmap and survival curvesin Figure 5. Note that for the heatmap, to prevent it from getting cluttered, we only show 60 rows in the heatmap even though there are a total of 190 rows after feature discretization (as mentioned in Section 3.1, we sort features largest to smallest based on the maximum minus minimum intensity value per row although we keep the same variable together, e.g., the different discretized values for "age" remain in sequence together).

From the heatmap in Figure 5(a), we can readily tease apart some trends. For instance, of the 5 largest clusters, the one with the lowest median survival time contains patients who are older and whose heart donors are also older. In this cluster, there are also many patients who are either overweight or obese (BMI value of 25 kg/m^2 or higher), which is a known risk factor for survival of cardiac transplant patients (Russo et al., 2010). The rightmost cluster with the highest median survival time contains mostly young patients with low weight, low BMI, and low donor age.

From the survival curves in Figure 5(b), we can tease out some trends among the five clusters shown. For example, the blue cluster (median survival time 6.25 years) is nearly always the worst off among the five clusters shown. Meanwhile, the purple cluster (median survival time 11.66 years) tends to be among the worst off initially (along with the blue cluster) but then has a higher survival probability than the other four clusters shown after around 9 years.



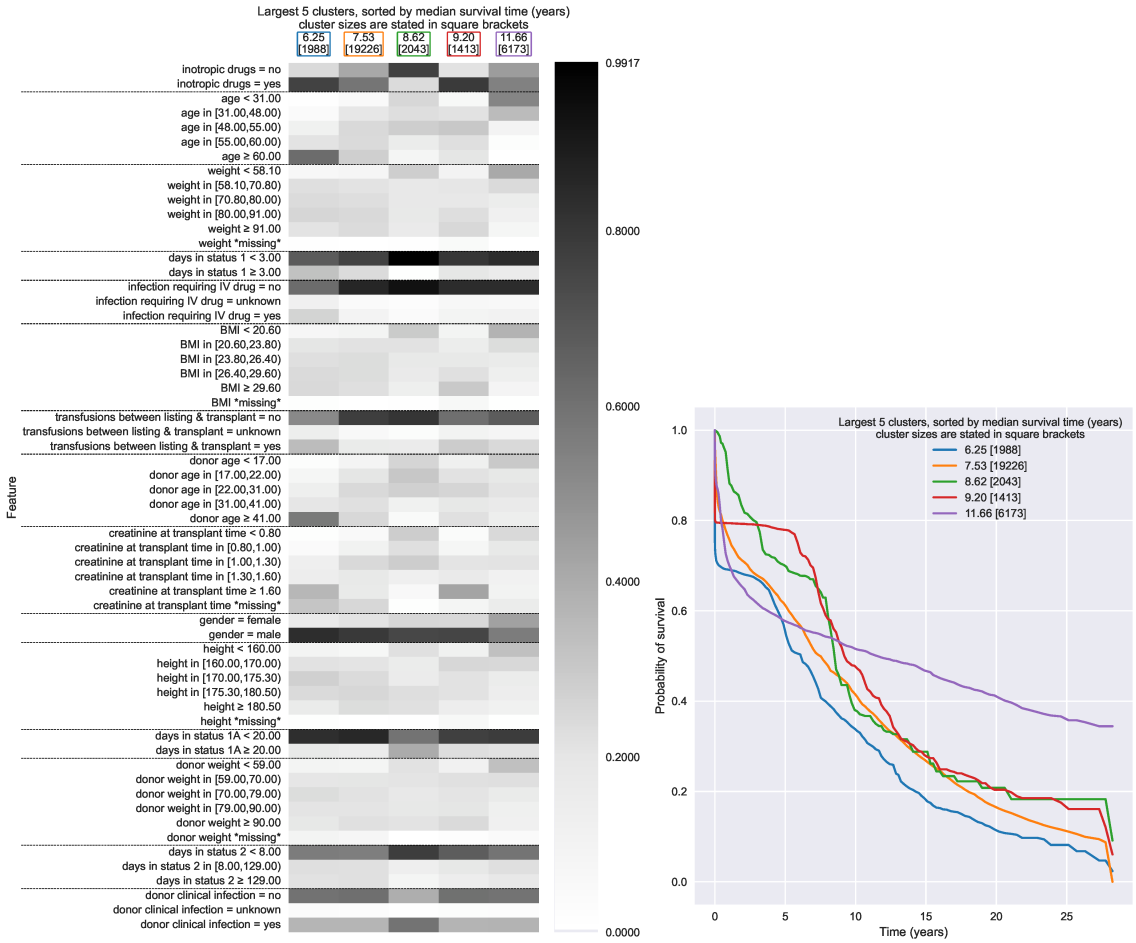
(a)



(b)

Figure 4: Visualization of all 10 clusters found by the final TUNA-KERNET (NO SPLIT, SFT) model trained on the ROTTERDAM/GBSG dataset (technically trained only on the Rotterdam portion of the data). Panel (a) shows a heatmap visualization that readily provides information on how the clusters are different, highlighting feature values that are prominent for specific clusters; the dotted horizontal lines separate features that correspond to the same underlying variable. Panel (b) shows survival curves (estimated from learned summary functions) for the same clusters as in panel (a); the x-axis indicates recurrence free survival time in months.

Visualizing all 30 clusters simultaneously would result in a raw feature heatmap and survival curve plot that are too cluttered. We again use complete-linkage agglomerative clustering, this time partitioning the 30 clusters into 10 superclusters, which we then visualize in Figure 6. The resulting feature heatmap in Figure 6(a) provides a more complete picture of how features vary across the proper training dataset compared to the earlier



(a) (b)

Figure 5: Visualization of the largest 5 clusters found by the final TUNA-KERNET (NO SPLIT, SFT) model trained on the UNOS dataset. Panel (a) shows a heatmap visualization that readily provides information on how the clusters are different, highlighting feature values that are prominent for specific clusters; the dotted horizontal lines separate features that correspond to the same underlying variable. Panel (b) shows survival curves (estimated from learned summary functions) for the same clusters as in panel (a); the x-axis measures the number of years since a patient received a heart transplant.

heatmap in Figure 5(a). Note that the rightmost supercluster (with the highest median survival time of 25.13 years) corresponds to a single patient who can be considered an outlier. The five superclusters/columns in the middle of the heatmap (with median survival times 5.08, 5.88, 7.87, 8.62, and 9.20 years) exhibit trends quite similar to what we already saw in Figure 5. On the other hand, the other five superclusters (with median survival times 0.01, 0.04, 15.74, 16.74, and 25.13 years) are all for young patients who receive heart transplants typically from young donors (in fact, we could use a finer-grain discretization of continuous features into more than 5 evenly spaced quantiles to reveal that many patients

in these clusters are pediatric patients). Meanwhile, the survival curves in Figure 6(b) show much larger differences compared to the earlier plot in Figure 5(b) that was only of the 5 largest clusters.

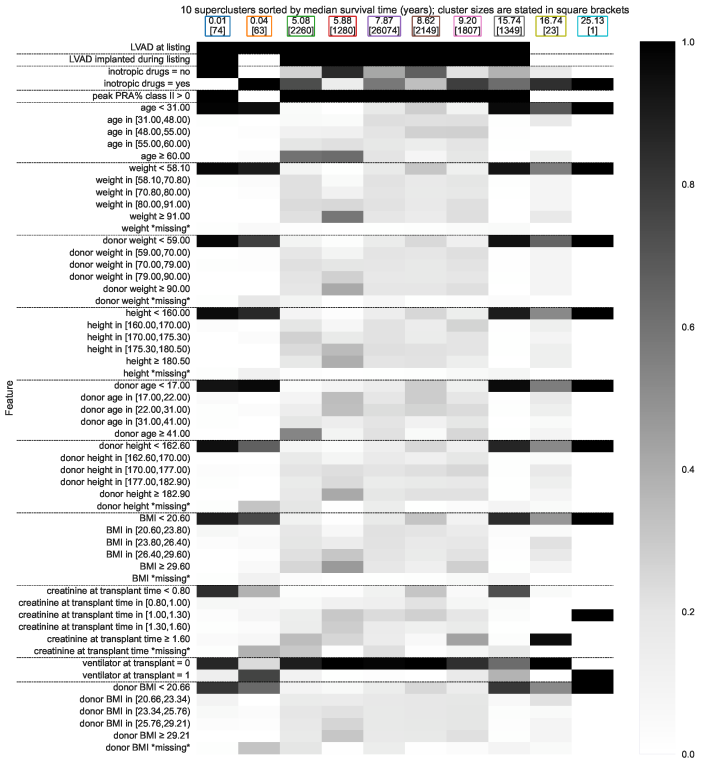
If we focus only on the five superclusters corresponding to young patients and re-plot the raw feature heatmap and survival curves only for these five superclusters, we obtain the plots in Figure 7. As a reminder, our strategy for ranking raw features in our heatmap visualization depends on which specific (super)clusters are being displayed, which is why the raw features are ranked differently between Figure 6(a) and Figure 7(a). From Figure 7(a), we see that the differences in these five superclusters could be explained in part by features such as whether a Left Ventricular Assist Device (LVAD) was already in place or implanted at the time of listing, whether inotropic drugs were used, creatinine level at transplant time, and whether a ventilator was used at transplant time; in fact, these features are known to be relevant for survival of pediatric cardiac transplantation (Dipchand, 2018).

KKBOX dataset We can repeat the same visualization ideas for the KKBOX dataset. As the ideas are the same, we provide only a summary of findings. Using the same supercluster idea presented previously, we visualize 10 superclusters in Figure 8 for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the KKBOX dataset. These superclusters summarize all 108,050 clusters found by the final TUNA-KERNET (NO SPLIT, SFT) model and contain 1,576,251 data points (the proper training data). We also provide a visualization of just the largest 10 clusters (containing 28.7% of the proper training data) in Appendix C.4. Among superclusters corresponding to users who subscribed to the music streaming service for less than a month, there's a higher fraction of users who are less than 19 years old and who have the lowest amount of payment. Among superclusters corresponding to the longest subscription times, there tends to be fewer previous churns for these users. Many of the survival curves show a steep drop in survival probability at around the 30-day mark, corresponding to a one-month promotion period.

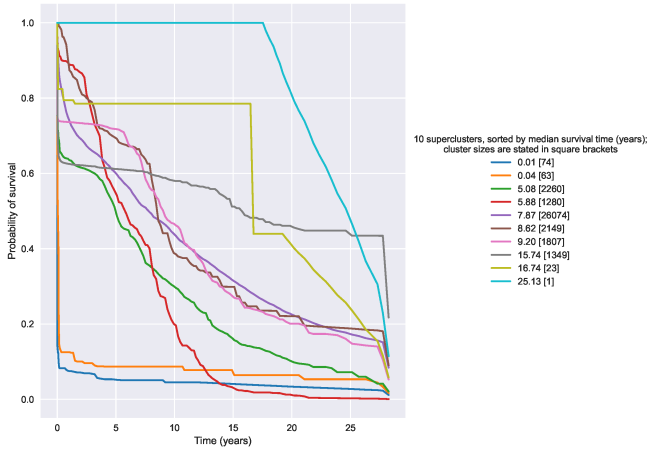
Final remarks on visualization We end this section with a reminder that as pointed out at the end of Section 3.1, a survival kernet model represents the hazard of any feature vector as a weighted combination of clusters. We can determine which clusters have nonzero weight for any given feature vector and only visualize these particular clusters. This visualization could be helpful to provide as "forecast evidence" or to assist model debugging. As an example, we can find test data with predictions that are inconsistent with the ground truth (e.g., if the test data point is not censored and its observed survival time is far from the predicted median survival time, or if the test data point is censored and the predicted median survival time is much lower than the observed time). For these test data that the model has difficulty with, we could examine which clusters have high weight, what features are prominent for these clusters, and what the clusters' survival curves are. After all, the predictions are made with these clusters' summary functions.

6. Discussion

In high-stakes applications such as healthcare, for survival models to be deployed in practice and producing time-to-event predictions using large (potentially live) streams of data in years to come, we believe that these models should be accurate, scalable, robust, inter-

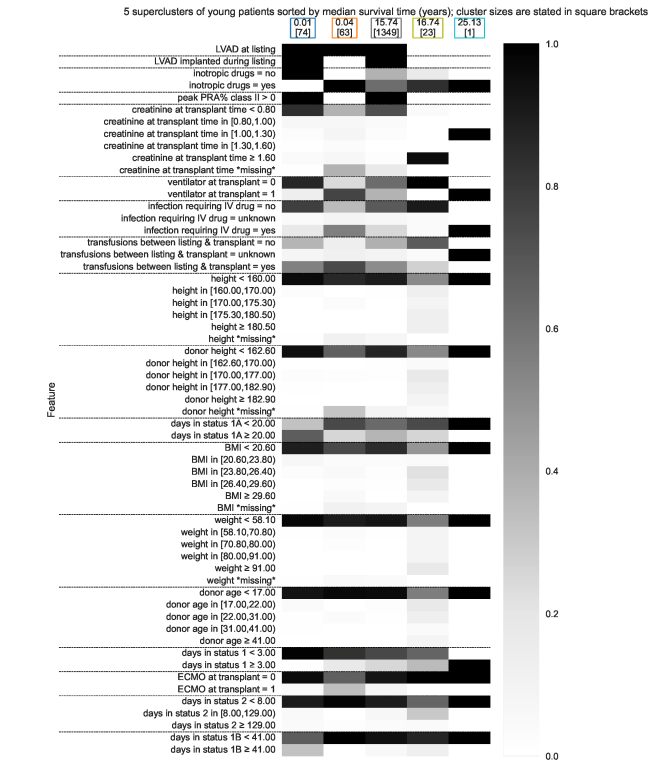


(a)

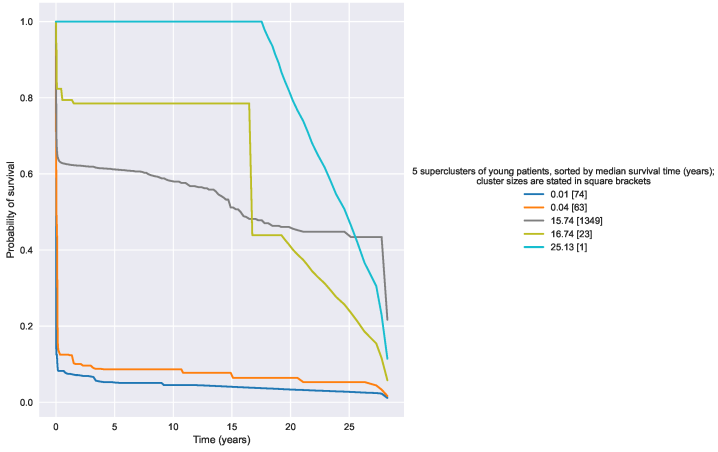


(b)

Figure 6: Visualization of 10 superclusters (summarizing 30 clusters) for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the UNOS dataset. Panel (a) shows a feature heatmap visualization (only 59 rows of features are shown out of 190). Panel (b) shows survival curves for the same superclusters as in panel (a); the x-axis measures the number of years since a patient received a heart transplant.

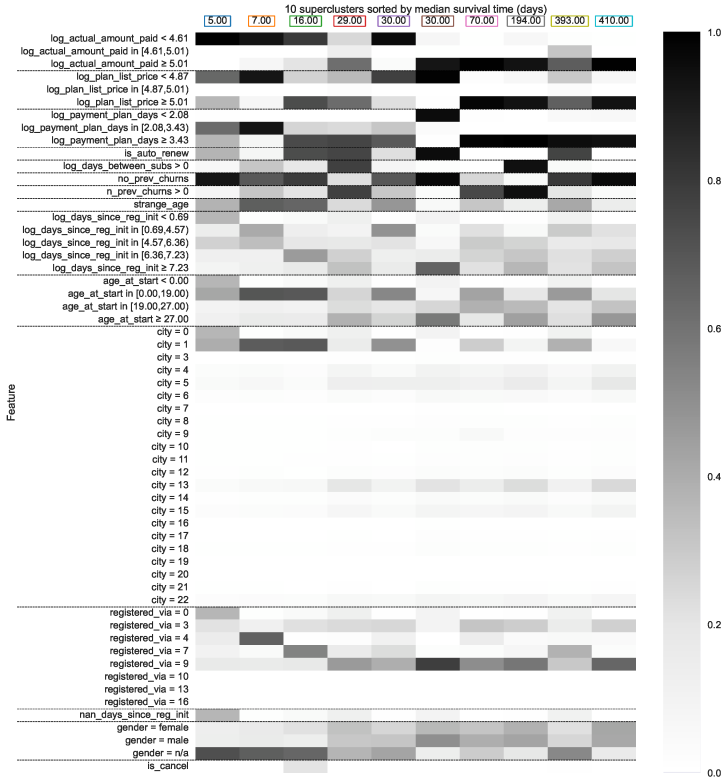


(a)

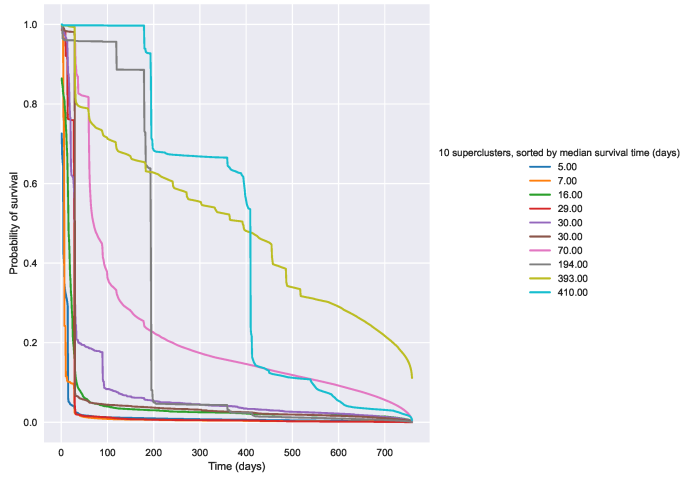


(b)

Figure 7: For the UNOS dataset, among the 10 superclusters from Figure 6, we now only display the superclusters with median survival times 0.01, 0.04, 15.74, 16.74, and 25.13 years corresponding to young patients. Panel (a) shows a feature heatmap visualization (only 59 rows of features are shown out of 190), which has the raw features sorted differently from Figure 6(a) to emphasize how these young patients' superclusters differ. Panel (b) shows survival curves for the same superclusters as in panel (a); as before, the x-axis measures the number of years since a patient received a heart transplant.



(a)



(b)

Figure 8: Visualization of 10 superclusters for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the KKBOX dataset. These 10 superclusters summarize all 108,050 clusters found by the TUNA-KERNET (NO SPLIT, SFT) model by merging clusters using complete-linkage agglomerative clustering. Panel (a) shows a feature heatmap visualization. Panel (b) shows survival curves for the same superclusters as in panel (a); the x-axis measures the number of days since an individual subscribed to the music streaming service.

pretable, and theoretically sound. Our proposed survival kernet model achieves many of these properties although some only partially. Specifically, survival kernets are accurate, scalable, interpretable, and for a special case of the model has a theoretical guarantee. We discuss a number of limitations of our work next.

Theoretical analysis of the best-performing survival kernet variant As our experimental results show, the best-performing variant of survival kernets sets pre-training and training data to be the same and also uses summary fine-tuning. However, neither of these are covered by our theoretical analysis.

From a technical standpoint, sample splitting (i.e., having pre-training and training data be disjoint and moreover appear i.i.d.) is crucial in our theoretical analysis. Specifically, our proof of Theorem 5 routinely uses the fact that the training embedding vectors $\widetilde{X}_1 = \widehat{\phi}(X_1), \ldots, \widetilde{X}_n = \widehat{\phi}(X_n)$ appear i.i.d. If the pre-training and training data are actually the same, then the learned base neural net $\widehat{\phi}$ is a function of $X_1, \ldots, X_n$ (along with their observed times and event indicators), so the embedding vectors are no longer guaranteed to be independent. We do not currently know of a good way to resolve this technical issue. We expect that a proof technique that can address this issue would be quite broadly applicable beyond analyzing survival kernets.[8]

A key reason for why our theoretical analysis requires sample splitting is that our analysis is agnostic to the choice of base neural network ϕ used. We leave the choice of ϕ up to the modeler since in practice, this is indeed how neural survival analysis models are used. For example, depending on the format of raw inputs, different neural net architectures could be used (e.g., using a multilayer perceptron for tabular data, using a convolutional neural network or vision transformer for images, using a recurrent neural network for variable-length time series). Even for the same class of base neural networks, such as multilayer perceptrons, the modeler could choose to try different options (such as whether to use batch norm). In contrast, existing neural survival analysis models with theoretical guarantees (Zhong et al., 2021, 2022) require the base neural network to be a multilayer perceptron with a number of constraints that are not typically used by practitioners (ℓ_0 and sup norm constraints) and, moreover, their theoretical guarantees assume that the neural network training finds the global minimum of the training loss, which is typically an impractical assumption.

As for summary fine-tuning, we suspect that if one treats the learned kernel function as fixed, then theoretical analysis should be possible although the theory would not say anything about the learned kernel function. Of course, the theory that we have presented does not say anything about the learned kernel function either since we treat the base neural net as a black box. Future work could consider the case when the base neural net is chosen from a specific family (e.g., a multilayer perceptron with ReLU activations), which could

8. The idea of using sample splitting to get guarantees really is not limited to our problem setup. Another example of using sample splitting has been in decision forest regression, where a commonly used proof technique (e.g., Biau 2012; Denil et al. 2014; Wager and Athey 2018; Athey et al. 2019)—that translates into how the decision trees in the forest are trained—is to split the training data into two portions: the first portion is used to decide on branching rules for the decision trees, whereas the second portion is used to decide on predicted labels at the leaves of the trees. In practice, random forests are rarely trained using this sort of sample splitting and, in fact, this sample splitting empirically can worsen the model's prediction accuracy in some circumstances (e.g., see the UCI dataset experiments by Denil et al. (2014)).

lead to a more nuanced theoretical guarantee (such as what is done by Zhong et al. (2021, 2022) for other deep survival analysis models, although as we already pointed out, their analysis requires some impractical assumptions on the neural net family).

A separate issue related to the theory is whether it is possible to come up with diagnostics that could be practically computed for any learned embedding space to assess to what extent it satisfies the theoretical assumptions relating the embedding space to survival and censoring times. As far as we are aware, there is no existing way to do this even if the neural net is the identity function so that the embedding space is just the raw feature space.

Better handling small datasets Survival kernets are inherently nonparametric. They make predictions using all the training data, possibly with some compression. We saw in the experimental results that survival kernets do not work as well as several baselines on the smallest dataset tested (ROTTERDAM/GBSG), for which one possible explanation is that for this dataset, the proportional hazards assumption is reasonable. A future research direction could be to figure out how to combine survival kernets with a parametric model so that when there are too few data points, we mostly use the parametric model (which could, for instance, use a proportional hazards assumption or an accelerated failure time assumption), and as more data become available, we gradually transition to taking a nonparametric approach. Figuring out a seamless way to do this transition could be an interesting research direction to explore.

Better handling low-dimensional structure The KKBOX dataset has a somewhat peculiar structure not present in the other datasets we considered: the observed survival times for KKBOX are always integers in the set $\{1, 2, \ldots, 759\}$; even though there are close to 3 million data points, the number of unique observed times is exactly 759. The number of unique observed times divided by the total number of data points is equal to 0.000270. In contrast, this ratio is significantly higher for all the other datasets considered: the ratio is equal to 0.551 for the ROTTERDAM/GBSG dataset, 0.193 for the SUPPORT dataset, and 0.149 for the UNOS dataset. Put another way, the KKBOX dataset has a significantly smaller time grid needed to represent observed times, which could be thought of as some low-dimensional structure in time. Of course, there could separately be low-dimensional structure present in the feature vectors themselves.

Currently, survival kernets do not have any sort of special procedure for exploiting low-dimensional structure. As already stated above, survival kernets are inherently nonparametric and compute predictions based on similarity scores to training points. If training feature vectors are noisy, then if we do not denoise the feature vectors somehow (e.g., using singular value thresholding as is done in principal component regression (Agarwal et al., 2021)), then we suspect that survival kernets could struggle to learn a good model compared to a simpler parametric model. While using an ε-net to cluster the training data can help denoise, right now we do not have an efficient manner that quickly determines what value of ε should be used. Moreover, this ε-net construction is only done after the base neural net ϕ has already been trained, which means relying on the ε-net to denoise would not affect neural net training itself (for which how we are currently learning the base neural net using the infinite-bandwidth kernel is not explicitly incorporating any denoising mechanism).

Improving the warm-start strategy and hyperparameter tuning Another possible explanation for why survival kernets does not achieve the best accuracy on the KKBOX

dataset could be that for KKBOX, the XGBOOST initialization is not very good. Note that even when using XGBOOST for survival analysis, there are a number of hyperparameters that can be set, and for simplicity, currently we are just using Cox regression as XGBOOST's objective. At the time of writing, XGBOOST also supports using accelerated failure time models with Gaussian, logistic, and extreme value distributions. Sweeping over more of these options could improve XGBOOST's prediction accuracy which could in turn improve the accuracy of a survival kernet model when using the TUNA warm-start procedure with XGBOOST.

Separately, as already pointed out by Chen (2020) in the original DKSA paper, using a tree ensemble to warm-start neural net training is not required. An alternative approach is to use the base neural net learned from, for instance, DEEPHIT (possibly removing a subset of the final layers of the neural net) and then fine-tuning using the DKSA loss.

Ultimately, although we have demonstrated that our TUNA warm-start strategy is clearly better than using standard random neural net parameter initialization, figuring out the best warm-start strategy to use for survival kernets (which might be dataset-dependent) remains an interesting open question. For example, one research direction could be to see whether contrastive representation learning (e.g., see the survey by Le-Khac et al. (2020)) could be used to warm-start survival kernet training and, if so, how well it works.

More generally, the TUNA warm-start strategy also aims to save time with hyperparameter selection by having some hyperparameters tuned as part of the warm-start procedure and then treated as fixed afterward. We leave a more thorough investigation of how to optimize hyperparameters for future work. For simplicity, when sweeping over hyperparameters, we used grid search. We did not explore other strategies such as Bayesian hyperparameter optimization (e.g., Akiba et al. 2019) nor did we carefully try to determine if some hyperparameters simply do not need to be tuned at all (in that there is a reasonable default value that can be used).

Impact of clustering on accuracy and interpretability For survival kernets, one could easily replace the ε-net-based clustering with another clustering approach. We chose the ε-net-based clustering for theoretical convenience: ε-nets can easily be constructed in the greedy manner we had mentioned, and importantly, the resulting clusters come with theoretical properties. Consequently, ε-nets have been used to prove that "compressed" versions of nearest neighbor or kernel regression are consistent (Kpotufe and Verma, 2017; Kontorovich et al., 2017; Hanneke et al., 2021). In contrast, had we used, for instance, k-means for clustering, then even though k-means at this point also has a lot of known theory (e.g., Rakhlin and Caponnetto 2006; Ben-David et al. 2007; Kumar and Kannan 2010; von Luxburg 2010; Fränti and Sieranoja 2019), it can be quite unstable in practice and the resulting clusters do not in general come with theoretical assurances. As such, we suspect that using k-means clustering with survival kernets would lead to a test-time predictor that is harder to theoretically analyze.

Putting aside theoretical convenience, by empirically evaluating how a wider range of clustering algorithms work with our survival kernet framework, we could potentially find that some of these lead to better prediction accuracy or better model interpretability compared to using ε-net clustering. We remark that especially as our experiments use embedding vectors that are on a hypersphere, then hyperspherical clustering methods could be used

such as fitting a mixture of von Mises-Fisher distributions (Banerjee et al., 2005). We leave an empirical study of using different clustering methods with survival kernets for future work.

Handling outliers Separately, we point out that even with our ε-net clustering approach, in practice many clusters could have very few data points assigned to them. Currently we are not removing such small clusters, although it might make sense to do so or to somehow flag these clusters as outliers and treat them a bit differently. Even pointing these clusters out to the user and visualizing them using the cluster visualization approach we had presented could be useful for model debugging purposes. In fact, this problem of addressing outliers would also arise if we replace the clustering method with, for instance, DBSCAN (Ester et al., 1996), which automatically flags various data points as outliers as part of the clustering procedure.

Calibration When a survival model's predicted number of critical events within a time interval closely resembles the observed number, then the model is considered well-calibrated (Haider et al., 2020). This could be thought of as a different notion of accuracy than the one we used in our experiments (namely time-dependent concordance index by Antolini et al. (2005)). We have not considered calibration in our paper, which in practice can be important. A straightforward way to incorporate calibration is to introduce an additional loss term such as the X-CAL loss by Goldstein et al. (2020). We leave a thorough investigation of the calibration properties of survival kernets, with and without the addition of the X-CAL loss, to future work.

Robustness We have not discussed robustness of survival kernets thus far although in some sense they already have a limited amount of robustness built-in. Specifically, survival kernets do not assume a parametric form for the underlying survival distribution; the base neural net is used only to parameterize the kernel function, which is then plugged into the nonparametric conditional Kaplan-Meier estimator. By being nonparametric, survival kernets should be more robust to the data coming from different survival distributions. In fact, our theoretical guarantee works under a fairly broad range of settings. However, survival kernets are currently not designed to handle changes in the data distribution such as covariate shift, where test feature vectors come from a different distribution than that of training but how feature vectors relate to survival and censoring times remains unchanged at test time. Modifying survival kernets to better accommodate test data appearing different from training data is an important future research direction to explore.

Acknowledgments

This work was supported in part by NSF CAREER award #2047981, and by Health Resources and Services Administration contract 234-2005-370011C. The content is the responsibility of the author alone and does not necessarily reflect the views or policies of the National Science Foundation or the Department of Health and Human Services, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government. We thank Shu Hu for pointing out some typos in an earlier draft, and we thank the anonymous reviewers for very helpful feedback.

Appendix A. Proofs

We present the proofs for Claim 3 (unit hypersphere's covering number) and Claim 4 (intrinsic dimension of the unit hypersphere) in Appendices A.1 and A.2, respectively. The proof of our main theoretical result Theorem 5 (survival kernet error bound) is in Appendix A.3.

A.1 Proof of Claim 3

Let $\{\zeta_1, \zeta_2, \ldots, \zeta_N\}$ be an $\varepsilon/2$-covering of the d-dimensional unit Euclidean ball (denoted by $\mathcal{B}_d$) such that N is the smallest value possible, i.e., $N = N(\varepsilon/2; \mathcal{B}_d)$; note that the unit Euclidean ball includes the interior of the ball whereas the unit hypersphere $\mathbb{S}^{d-1}$ is only the outer shell. We recall a standard result of covering numbers (Corollary 4.2.13 of Vershynin (2018)) that

$$N = N(\varepsilon/2; \mathcal{B}_d) \leq \Big(\frac{4}{\varepsilon} + 1\Big)^d. \tag{A.1}$$

Then since $\mathbb{S}^{d-1} \subseteq \mathcal{B}_d$, for every $v \in \mathbb{S}^{d-1}$, there exists some ζ_j such that $\|v - \zeta_j\| \leq \varepsilon/2$. This implies that there exists a subset Ξ of $\{\zeta_1, \zeta_2, \ldots, \zeta_N\}$ such that every point in $\mathbb{S}^{d-1}$ is at most a distance $\varepsilon/2$ from some point in Ξ. Let Ξ^* be such a subset that has the smallest size, and denote its unique elements as $\xi_1, \xi_2, \ldots, \xi_{|\Xi^*|}$.

We next show how to construct an ε-cover for $\mathbb{S}^{d-1}$ using $\xi_1, \xi_2, \ldots, \xi_{|\Xi^*|}$. For each point ξ_i, note that there exists some $s_i \in \mathbb{S}^{d-1}$ such that $s_i \in B(\xi_i, \varepsilon/2)$ (if such an s_i does not exist, then ξ_i would not have been included in Ξ^*, i.e., Ξ^* does not actually have the smallest size possible while still covering $\mathbb{S}^{d-1}$). Next, we observe that for any $v \in B(\xi_i, \varepsilon/2)$, the triangle inequality implies that

$$\|s_i - v\|_2 = \|s_i - \xi_i + \xi_i - v\|_2 \leq \underbrace{\|s_i - \xi_i\|_2}_{\leq \varepsilon/2} + \underbrace{\|\xi_i - v\|_2}_{\leq \varepsilon/2},$$

which means that v is also in $B(s_i, \varepsilon)$. Hence,

$$B(s_i, \varepsilon) \supseteq B(\xi_i, \varepsilon/2).$$

This means that $\cup_{i=1}^{|\Xi^*|} B(s_i, \varepsilon) \supseteq \cup_{i=1}^{|\Xi^*|} B(\xi_i, \varepsilon/2)$, and since the latter covers $\mathbb{S}^{d-1}$, so does the former. We thus conclude that $\{s_1, s_2, \ldots, s_{|\Xi^*|}\}$ forms a valid ε-cover of $\mathbb{S}^{d-1}$. The ε-cover of $\mathbb{S}^{d-1}$ that has the smallest size possible is thus upper-bounded by $|\Xi^*| \leq N \leq \big(\frac{4}{\varepsilon} + 1\big)^d$ using inequality (A.1). ■

A.2 Proof of Claim 4

Let $f_{\widetilde{\mathsf{X}}}$ denote the PDF of $\mathbb{P}_{\widetilde{\mathsf{X}}}$, where we assume that $f_{\widetilde{\mathsf{X}}}(\widetilde{z}) \geq c_{\min} > 0$ for all $\widetilde{z} \in \mathbb{S}^{d-1}$, and $f_{\widetilde{\mathsf{X}}}(\widetilde{z}) = 0$ for all $\widetilde{z} \notin \mathbb{S}^{d-1}$. Let $\widetilde{x} \in \widetilde{\mathcal{X}}$ and $r \in (0, 1]$. Then

$$\mathbb{P}_{\widetilde{\mathsf{X}}}(B(\widetilde{x}, r)) = \int_{B(\widetilde{x}, r)} f_{\widetilde{\mathsf{X}}}(\widetilde{z}) d\widetilde{z} \geq \int_{B(\widetilde{x}, r)} c_{\min} \cdot d\widetilde{z} = c_{\min} \cdot \mathrm{area}\big(\underbrace{\mathbb{S}^{d-1} \cap B(\widetilde{x}, r)}_{\mathcal{C}(\widetilde{x}, r) :=}\big).$$

Note that $\mathcal{C}(\widetilde{x}, r)$ is a hyperspherical cap. We make a geometric observation:

$$\mathrm{area}\big(\mathcal{C}(\widetilde{x}, r)\big) \geq \mathrm{volume}\big(\underbrace{\mathcal{C}(\widetilde{x}, r) \text{ projected down to } \mathbb{R}^{d-1} \text{ in the direction of } \widetilde{x}}_{\mathcal{A} :=}\big),$$

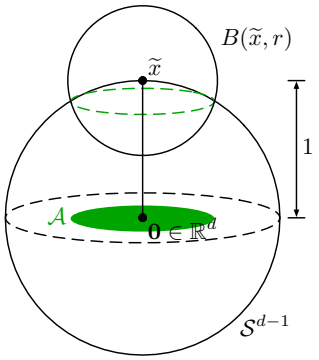


Figure 9: This diagram shows the geometric observation in the proof for the case of $d = 3$ dimensional Euclidean space. The intersection of the closed ball $B(\widetilde{x}, r)$ with the unit hypersphere $\mathbb{S}^{d-1}$ is a hyperspherical cap, with the "top" of the cap at $\widetilde{x}$, and the bottom is shown in the dotted green line. The surface area of this cap is lower-bounded by the area of the projection (the solid green circle $\mathcal{A}$; in higher dimensions, this is a ball in $\mathbb{R}^{d-1}$).

where $\mathcal{A}$ is depicted as a solid shaded green circle in Figure 9. The core of the remaining analysis is in determining the volume of $\mathcal{A}$. With a bit of trigonometry, we can derive that $\mathcal{A}$ is precisely equal to a ball centered at the origin with radius $\frac{1}{2}r\sqrt{4-r^2}$; see Figure 10. Since $r \in (0, 1]$, the radius of $\mathcal{A}$ satisfies the bound $\frac{1}{2}r\sqrt{4-r^2} \geq \frac{1}{2}r\sqrt{3}$. Thus, the volume of $\mathcal{A}$ is lower-bounded by the volume of a ball in $\mathbb{R}^{d-1}$ with radius $\frac{\sqrt{3}}{2}r$, which is $\frac{\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2}+1)}(\frac{\sqrt{3}}{2}r)^{d-1}$, where $\Gamma(z) := \int_0^\infty u^{z-1}e^{-u}du$. Putting everything together,

$$\begin{aligned}
\mathbb{P}_{\widetilde{\mathsf{X}}}(B(\widetilde{x}, r)) &\geq c_{\min} \cdot \text{area}(\mathcal{C}(\widetilde{x}, r)) \\
&\geq c_{\min} \cdot \text{volume}(\mathcal{A}) \\
&\geq c_{\min} \cdot \frac{\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2}+1)}\Big(\frac{\sqrt{3}}{2}r\Big)^{d-1} \\
&= \Big[c_{\min} \cdot \frac{\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2}+1)}\Big(\frac{\sqrt{3}}{2}\Big)^{d-1}\Big] r^{d-1},
\end{aligned}$$

which establishes that the embedding space has intrinsic dimension $d - 1$.

A.3 Proof of Survival Kernet Error Bound (Theorem 5)

To keep the notation from getting cluttered, we use different notation than what is in the main paper. We drop tildes and instead write the embedding space as $\mathcal{U} \subset \mathbb{R}^d$. The training (and not pre-training) embedding vectors (treated as random variables) are written as $U_1, U_2, \ldots, U_n \in \mathcal{U}$. Just as in the main paper, we use the notation $U_{1:n} = (U_1, \ldots, U_n) \in \mathcal{U}^n$. The random test data point's embedding vector is denoted by the random variable U. The marginal distribution of embedding vectors is denoted by $\mathbb{P}_{\mathsf{U}}$. We denote the ε-net as Q (treated as a set), which is a subsample of the indices in $U_{1:n}$ (treated as a sequence). Just as in the main paper, we denote $\mathcal{I}_q \subseteq \{1, 2, \ldots, n\}$ to be the indices of the training points assigned to $q \in Q$. We denote the true conditional survival, censoring, and observed

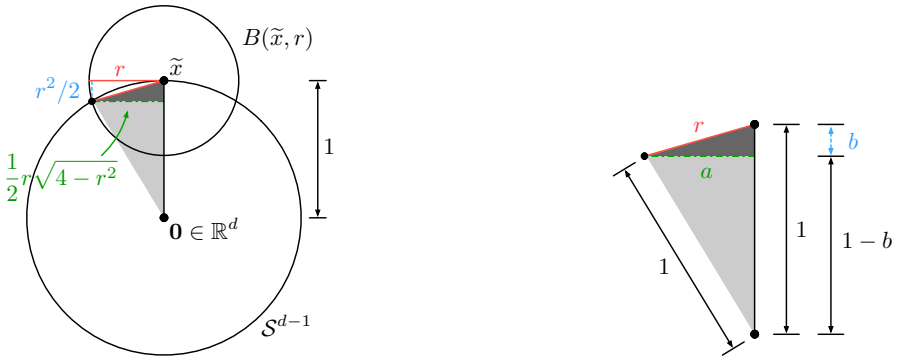


Figure 10: The radius of both the dotted and solid green circles in Figure 9 can be computed to be equal to $\frac{1}{2}r\sqrt{4-r^2}$. Specifically, in panel (a), we show a 2D view of Figure 9. To verify that the lengths specified in red, blue, and green in panel (a) are correct, we show the two shaded right triangles from panel (a) with more detail in panel (b), where we denote the green and blue lengths as a and b respectively: the Pythagorean theorem says that for the darker triangle, we have $a^2+b^2=r^2$, and for the lighter triangle, we have $a^2+(1-b)^2=1^2$; by solving this system of two equations, we get $a=\frac{1}{2}r\sqrt{4-r^2}$ and $b=\frac{r^2}{2}$.

time distributions as $\mathbb{P}_{\mathsf{T}|\mathsf{U}}$, $\mathbb{P}_{\mathsf{C}|\mathsf{U}}$, and $\mathbb{P}_{\mathsf{Y}|\mathsf{U}}$. We write their PDFs as $f_{\mathsf{T}|\mathsf{U}}(t|u)$, $f_{\mathsf{C}|\mathsf{U}}(t|u)$, and $f_{\mathsf{Y}|\mathsf{U}}(t|u)$; their CDFs as $F_{\mathsf{T}|\mathsf{U}}(t|u)$, $F_{\mathsf{C}|\mathsf{U}}(t|u)$, and $F_{\mathsf{Y}|\mathsf{U}}(t|u)$; and their tail functions as $S_{\mathsf{T}|\mathsf{U}}(t|u)=1-F_{\mathsf{T}|\mathsf{U}}(t|u)$, $S_{\mathsf{C}|\mathsf{U}}(t|u)=1-F_{\mathsf{C}|\mathsf{U}}(t|u)$, and $S_{\mathsf{Y}|\mathsf{U}}(t|u)=1-F_{\mathsf{Y}|\mathsf{U}}(t|u)$. For a test embedding vector u, the conditional survival function we aim to predict is precisely $S_{\mathsf{T}|\mathsf{U}}(t|u)$ for $t \geq 0$.

For survival kernets, the only training points that can possibly contribute to the prediction for test embedding vector u (i.e., they have nonzero truncated kernel weight) are the ones with indices in

$$\mathcal{N}_Q(u) := \coprod_{q \in Q \text{ s.t. } \|u-q\|_2 \leq \tau} \mathcal{I}_q,$$

where "$\coprod$" denotes disjoint union. This union is disjoint since the survival kernet training procedure assigns each training point to exactly one exemplar in Q. The set $\mathcal{N}_Q(u)$ could be thought of as the "neighbors" of u.

For $Y_{\max}(u) := \max_{j \in \mathcal{N}_Q(u)} Y_j$, the survival kernet hazard function estimate (equation (3.3)) can be written as follows, using the new notation:

$$\widehat{g}_Q(t|u) := \begin{cases} \dfrac{\sum_{j \in \mathcal{N}_Q(u)} K(\|u-q_j\|_2) D_j \mathbb{1}\{Y_j = t\}}{\sum_{j \in \mathcal{N}_Q(u)} K(\|u-q_j\|_2) \mathbb{1}\{Y_j \geq t\}} \\ \qquad \text{if } \mathcal{N}_Q(u) \neq \emptyset \text{ and } 0 \leq t \leq Y_{\max}(u), \\ 0 \qquad \text{otherwise.} \end{cases} \tag{A.2}$$

Let $\mathcal{T}$ be the set of (unique) times until death within the training data, i.e., $\mathcal{T} := \{t' \in [0,\infty) : \exists j \in \{1,\ldots,n\} \text{ s.t. } Y_j = t' \text{ and } D_j = 1\}$. Then the survival kernet conditional

survival function estimator can be written as

$$\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) := \prod_{t' \in \mathcal{T}} (1 - \widehat{g}_Q(t'|u))^{\mathbb{1}\{t' \leq t\}}.$$

In the main paper, this estimator is written as $\widetilde{S}_{\widetilde{\mathcal{Q}}_\varepsilon}(t|\widetilde{x})$. Using the new notation, we state the survival kernet error guarantee (Theorem 5) in the following proposition (note that this proposition is more detailed than Theorem 5 and includes constants that we have not tried to optimize).

Proposition 6 *Suppose that assumptions* $\mathbf{A}^{technical}$, $\mathbf{A}^{intrinsic}$, *and* $\mathbf{A}^{survival}$ *hold, and that for training a survival kernet, we choose* $\varepsilon = \beta\tau$ *when constructing the* ε*-net* Q *using* $U_{1:n}$, *where* $\beta \in (0,1)$ *and* $\tau > 0$ *are user-specified parameters. Let*

$$\Psi = \min\left\{ N\Big(\frac{(1-\beta)\tau}{2}; \widetilde{\mathcal{X}}\Big), \frac{1}{C_{d'}((1-\beta)\tau)^{d'}} \right\}.$$

Then for

$$n \geq \frac{2}{C_{d'}((1-\beta)\tau)^{d'}} \max\left\{ \frac{K(0)}{K(\tau)\theta}, \Big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\Big)^{1/3} \right\},$$

we have

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n}, D_{1:n}, U_{1:n}, U}\Big[\frac{\int_0^{t_{horizon}} (\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|U) - S_{\mathsf{T}|\mathsf{U}}(t|U))^2 dt}{t_{horizon}}\Big] \\
&\leq \frac{1}{n} \cdot \frac{346K^4(0)\log(n^3 \frac{144K^2(\tau)}{9K^2(0)})}{\theta^4 K^4(\tau)} \Psi \\
&\quad + (1+\beta)^{2\alpha}\tau^{2\alpha}\Bigg[\frac{t_{horizon}^2}{\theta^2}\Big(\lambda_{\mathsf{T}}^2 + \frac{3\lambda_{\mathsf{T}} f_{\mathsf{T}}^* \lambda_{\mathsf{C}} t_{horizon}}{4} + \frac{3(f_{\mathsf{T}}^*)^2 \lambda_{\mathsf{C}}^2 t_{horizon}^2}{20}\Big) \\
&\qquad\qquad + \frac{12K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}} + \lambda_{\mathsf{C}})^2 t_{horizon}^2\Bigg],
\end{aligned}$$

where

$$f_{\mathsf{T}}^* := \max_{s \in [0, t_{horizon}], u \in \mathcal{U}} f_{\mathsf{T}|\mathsf{U}}(s|u).$$

In our theoretical analysis, we assume that the ε-net Q is a random variable that depends on $U_{1:n}$, where if we condition on $U_{1:n}$, then Q becomes deterministic. Note that as stated previously, we treat $U_{1:n}$ as a sequence (so ordering matters) whereas Q is treated as a set. Thus, an algorithm that constructs Q from $U_{1:n}$ need not produce the exact same Q if the ordering of $U_{1:n}$ is shuffled. If a randomized algorithm is used to construct Q, we assume that it has a random seed that can be fixed, making the algorithm behave deterministically given $U_{1:n}$.

A key lemma used in proving Proposition 6 is a lower bound on the number of neighbors of a test embedding vector u.

Lemma 7 (Number of training points contributing to prediction – lower bound) *Let $\beta \in (0,1)$. Suppose that we train a survival kernet with threshold distance $\tau > 0$ and compression parameter $\varepsilon = \beta\tau$, i.e., training embedding vectors $U_{1:n}$ are used to construct a $\beta\tau$-net Q. For a fixed choice of embedding vector $u \in \mathcal{U}$, let $U_{1:n}(u;\beta,\tau)$ denote the multiset of training data that land within the closed ball $B(u,(1-\beta)\tau)$. We have the inclusion relationship*

$$U_{1:n}(u;\beta,\tau) \subseteq \bigcup_{j \in \mathcal{N}_Q(u)} \underbrace{[U_j]}_{\text{multiset with single element}}. \tag{A.3}$$

In particular, taking the cardinality of both sides, we have the inequality

$$\Xi_u := |U_{1:n}(u;\beta,\tau)| \leq |\mathcal{N}_Q(u)|. \tag{A.4}$$

Proof Let $u' \in U_{1:n}(u;\beta,\tau)$. Since $U_{1:n}(u;\beta,\tau) = U_{1:n} \cap B(u,(1-\beta)\tau)$, we have $\|u'-u\|_2 \leq (1-\beta)\tau$. Next, let $q \in Q$ be the exemplar point that u' is assigned to by step 4 of the survival kernet training procedure. By how step 4 works, we have $\|u'-q\|_2 \leq \beta\tau$. We can then bound the distance between q and u using the triangle inequality:

$$\|q-u\|_2 \leq \underbrace{\|q-u'\|_2}_{\leq \beta\tau} + \underbrace{\|u'-u\|_2}_{\leq (1-\beta)\tau} \leq \tau.$$

This means that u' is a training embedding vector that is assigned to an exemplar point q that satisfies $\|u-q\|_2 \leq \tau$, i.e., u' corresponds to a training point in $\mathcal{N}_Q(u)$. This establishes the subset relationship (A.3). ■

Throughout our theoretical analysis, we assume that Q is constructed to be an ε-net of $U_{1:n}$ with $\varepsilon = \beta\tau$ just as in Lemma 7, i.e., $\beta \in (0,1)$ and $\tau > 0$ is the threshold distance for the truncated kernel.

We also regularly use the variable Ξ_u defined in equation (A.4). Importantly, since $U_{1:n}$ are sampled i.i.d. from $\mathbb{P}_{\mathsf{U}}$, then Ξ_u is binomially distributed with parameters n and $\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))$. We can ensure that test embedding vector u has enough neighbors with high probability by asking that Ξ_u be sufficiently large (for which Lemma 7 guarantees that $|\mathcal{N}_Q(u)| \geq \Xi_u$). Specifically, define the good event

$$\mathcal{E}_\beta(u) := \Big\{ \Xi_u > \frac{1}{2} n \mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau)) \Big\}. \tag{A.5}$$

We denote the complement of this event as $\mathcal{E}_\beta^c(u)$.

Lemma 8 (Sufficiently many number of neighbors) *Suppose that Assumption $\mathbf{A}^{technical}$ holds and $U_{1:n}$ are sampled i.i.d. from $\mathbb{P}_{\mathsf{U}}$. Let $\beta \in (0,1)$, $\tau > 0$, and $u \in \mathcal{U}$. Then the probability that good event $\mathcal{E}_\beta(u)$ does not happen is bounded above as*

$$\mathbb{P}(\mathcal{E}_\beta^c(u)) \leq \exp\Big(-\frac{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}{8} \Big) \leq \frac{8}{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}.$$

Proof The claim readily follows from a multiplicative Chernoff bound of the binomial distribution and noting that $\exp(-z) \leq 1/z$ for all $z > 0$. ■

Another argument used in the proof is that if the j-th training point is a neighbor of embedding vector u—i.e., $j \in \mathcal{N}_Q(u)$—then their embedding vectors cannot be too far apart, i.e., $\|U_j - u\|_2$ is not too large.

Lemma 9 (Neighbor embedding vectors are close) *Suppose that we train a survival kernet as in Lemma 7. Let $u \in \mathcal{U}$. If $j \in \mathcal{N}_Q(u)$, then*

$$\|U_j - u\|_2 \leq (1+\beta)\tau.$$

Proof Let $j \in \mathcal{N}_Q(u)$. This means that the exemplar that the j-th training point is assigned to, namely $q_j \in Q$, must be within distance τ of u. Thus,

$$\|U_j - u\|_2 \leq \underbrace{\|U_j - q_j\|_2}_{\leq \beta\tau} + \underbrace{\|q_j - u\|_2}_{\leq \tau} \leq (1+\beta)\tau,$$

where we have used the fact that Q is a $\beta\tau$-net, and $\mathcal{N}_Q(u)$ is defined to consider all exemplars within distance τ of u. ■

We remark that proving the special case when $\varepsilon = 0$ is easier precisely because Lemmas 7, 8, and 9 simplify — we no longer need ε-net arguments. Instead of Lemma 7, we directly consider training points within distance τ of fixed embedding vector u. Instead of Lemma 8, we directly consider points landing in the ball $B(u,\tau)$. In Lemma 9, there is no multiplicative approximation error $(1+\beta)$ to worry about. In short, we get the same conclusions as these lemmas except plugging in $\beta = 0$.

We present proof of Proposition 6 next.

A.3.1 Proof of Proposition 6

We focus on proving "pointwise" bounds first, which in this case means that we fix both the test embedding vector $u \in \mathcal{U}$ and time $t \in [0, t_{\text{horizon}}]$. (At the end of the proof, we take an expectation over the test embedding vector and integrate over time.)

First off, we have the trivial bound $|\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u)| \leq 1$. Under some good events holding, we can obtain a nontrivial error bound. The first good event is the same as for regression $\mathcal{E}_\beta(u) = \{\Xi_u > \frac{1}{2} n \mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\}$. We introduce two additional good events. We define the second good event as

$$\mathcal{E}_{\text{horizon}}(u) := \Big\{ \underbrace{\sum_{j \in \mathcal{N}_Q(u)} \mathbb{1}\{Y_j > t_{\text{horizon}}\}}_{d^+(u, t_{\text{horizon}}) :=} > \frac{|\mathcal{N}_Q(u)|\theta}{2} \Big\}. \tag{A.6}$$

Note that for any $s \geq 0$, $d^+(u,s)$ is the number of neighbors of u that have observed times Y_j's exceeding s. When event $\mathcal{E}_{\text{horizon}}(u)$ holds, we have $t_{\text{horizon}} \leq Y_{\max}(u)$. We define the third and final good event later when we relate estimating the tail function $S_{\mathsf{T}|\mathsf{U}}(t|u)$ (that is for survival times T_j's that we do not always get to observe in the training data), to estimating the tail function $S_{\mathsf{Y}|\mathsf{U}}(t|u)$ (that is of observed times Y_j's that we see all of in the training data).

Importantly, when both $\mathcal{E}_\beta(u)$ and $\mathcal{E}_{\text{horizon}}(u)$ hold, then for every time $t \in [0, t_{\text{horizon}}]$, the hazard estimator $\widehat{g}_Q(t|u)$ takes on a nontrivial value given by the first case of equation (A.2), rather than just being 0. As we show later (Lemma 14), good events $\mathcal{E}_\beta(u)$ and $\mathcal{E}_{\text{horizon}}(u)$ simultaneously hold with high probability as n grows large. These good events holding enable us to write the survival kernet conditional survival function estimate $\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u)$ in a form that will be easier to work with.

Lemma 10 (Convenient form of the kernel survival estimator) *Suppose that Assumptions* $\mathbf{A}^{technical}$ *and* $\mathbf{A}^{survival}$ *hold, and that the kernel function* K *is of the form described in equation* (3.1). *Consider training a survival kernet with parameter* $\varepsilon = \beta\tau$, *where* $\tau > 0$ *is the truncated kernel threshold distance, and* $\beta \in (0,1)$ *is a user-specified parameter. Fix* $u \in \mathcal{U}$, *and condition on events* $\mathcal{E}_\beta(u)$ *and* $\mathcal{E}_{horizon}(u)$ *occurring. Recall that* $q_j \in Q$ *is the exemplar that the* j*-th training point is assigned to during survival kernet training. With probability 1, for all* $t \in [0, t_{horizon}]$,

$$\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) = \prod_{i \in \mathcal{N}_Q(u)} \Big(\frac{d_K^+(u, Y_i)}{K_i + d_K^+(u, Y_i)} \Big)^{D_i \mathbb{1}\{Y_i \le t\}}, \tag{A.7}$$

where $K_i := K(\|u - q_i\|_2)\mathbb{1}\{\|u - q_i\|_2 \le \tau\}$, *and for any* $s \ge 0$,

$$d_K^+(u, s) := \sum_{j \in \mathcal{N}_Q(u)} K_j \mathbb{1}\{Y_j > s\}.$$

Moreover, since $\mathcal{E}_{horizon}(u) = \{d^+(u, t_{horizon}) > \frac{|\mathcal{N}_Q(u)|\theta}{2}\}$ *holds, we are guaranteed that* $\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) > 0$ *for* $t \in [0, t_{horizon}]$.

Note that $d_K^+(u,s)$ is not the same as $d^+(u,s)$ defined earlier (in equation (A.6), which does not use kernel weights). These quantities are related via the bound $d_K^+(u,s) \ge K(\tau) d^+(u,s)$.

Proof Fix $u \in \mathcal{U}$ and $t \in [0, t_{\text{horizon}}]$. First off, due to events $\mathcal{E}_\beta(u)$ and $\mathcal{E}_{\text{horizon}}(u)$ holding, $\widehat{g}_Q(t|u)$ is nonzero. Next, Assumption $\mathbf{A}^{\text{survival}}$ implies that ties in Y_j's happen with probability 0. Thus, with probability 1, there are no ties in Y_j's, so

$$\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) = \prod_{t' \in \mathcal{T}} (1 - \widehat{g}_Q(t'|u))^{\mathbb{1}\{t' \le t\}} = \prod_{i=1}^n (1 - \widehat{g}_Q(Y_i|u))^{D_i \mathbb{1}\{Y_i \le t\}}. \tag{A.8}$$

Next, since there are no ties in Y_j's, the expression for $\widehat{g}_Q(Y_i|u)$ simplifies as

$$\widehat{g}_Q(Y_i|u) = \frac{K_i D_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j \mathbb{1}\{Y_j \ge Y_i\}} = \frac{K_i D_i}{K_i + \sum_{j \in \mathcal{N}_Q(u)} K_j \mathbb{1}\{Y_j > Y_i\}} = \frac{K_i D_i}{K_i + d_K^+(u, Y_i)}.$$

Hence,

$$\begin{aligned}
(1 - \widehat{g}_Q(Y_i|u))^{D_i \mathbb{1}\{Y_i \le t\}} &= \Big(1 - \frac{K_i D_i}{K_i + d_K^+(u, Y_i)}\Big)^{D_i \mathbb{1}\{Y_i \le t\}} \\
&= \Big(1 - \frac{K_i}{K_i + d_K^+(u, Y_i)}\Big)^{D_i \mathbb{1}\{Y_i \le t\}} \\
&= \Big(\frac{d_K^+(u, Y_i)}{K_i + d_K^+(u, Y_i)}\Big)^{D_i \mathbb{1}\{Y_i \le t\}}.
\end{aligned}$$

Combining this with equation (A.8), we get

$$\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) = \prod_{i=1}^n \Big(\frac{d_K^+(u, Y_i)}{K_i + d_K^+(u, Y_i)} \Big)^{D_i \mathbb{1}\{Y_i \le t\}}.$$

In the RHS, if $K_i = 0$, then the i-th factor is 1 and thus does not affect the overall product. Note that $K_i > 0$ precisely for $i \in \mathcal{N}_Q(u)$. Thus, it suffices to only take the product over $i \in \mathcal{N}_Q(u)$, which establishes equality (A.7). ■

When we can write $\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u)$ using the expression (A.7), then using the Taylor expansion $\log(1+z) = \sum_{\ell=1}^{\infty} \frac{1}{\ell}(\frac{z}{z+1})^{\ell}$ that is valid for any $z > 0$, we get

$$\begin{aligned}
\log \widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) &= \sum_{i \in \mathcal{N}_Q(u)} D_i \mathbb{1}\{Y_i \le t\} \log\Big(\frac{d_K^+(u, Y_i)}{K_i + d_K^+(u, Y_i)}\Big) \\
&= -\sum_{i \in \mathcal{N}_Q(u)} D_i \mathbb{1}\{Y_i \le t\} \log\Big(1 + \frac{K_i}{d_K^+(u, Y_i)}\Big) \\
&= -\sum_{i \in \mathcal{N}_Q(u)} \frac{D_i \mathbb{1}\{Y_i \le t\} K_i}{K_i + d_K^+(u, Y_i)} - \sum_{i \in \mathcal{N}_Q(u)} D_i \mathbb{1}\{Y_i \le t\} \sum_{\ell=2}^{\infty} \frac{1}{\ell[1 + (d_K^+(u, Y_i)/K_i)]^{\ell}} \\
&= W_1(t|u) + W_2(t|u) + W_3(t|u),
\end{aligned}$$

where

$$\begin{aligned}
W_1(t|u) &:= \sum_{i \in \mathcal{N}_Q(u)} \Big(\frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big)\Big(-\frac{D_i \mathbb{1}\{Y_i \le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big), \\
W_2(t|u) &:= -\sum_{i \in \mathcal{N}_Q(u)} \frac{K_i D_i \mathbb{1}\{Y_i \le t\}\big[\frac{\sum_{\ell \in \mathcal{N}_Q(u)} K_\ell}{K_i + d_K^+(u, Y_i)} - \frac{1}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\big]}{\sum_{j \in \mathcal{N}_Q(u)} K_j}, \\
W_3(t|u) &:= -\sum_{i \in \mathcal{N}_Q(u)} D_i \mathbb{1}\{Y_i \le t\} \sum_{\ell=2}^{\infty} \frac{1}{\ell[1 + (d_K^+(u, Y_i)/K_i)]^{\ell}}.
\end{aligned}$$

The high-level idea is to show that $W_1(t|u) \to \log S_{\mathsf{T}|\mathsf{U}}(t|u)$, $W_2(t|u) \to 0$, and $W_3(t|u) \to 0$.

As previously pointed out by Chen (2019), the term $W_1(t|u)$ could be thought of as a kernel regression estimate that averages over hypothetical labels of the form $-\frac{D_i \mathbb{1}\{Y_i \le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}$ across i, which knows the true tail function $S_{\mathsf{Y}|\mathsf{U}}$ (of observed times and not survival times). Of course this true tail function is not actually known.

The term $W_2(t|u)$ focuses on how accurately we can estimate the tail function $S_{\mathsf{Y}|\mathsf{U}}$. Note that estimating the CDF of the observed times Y_j's is simpler than estimating the CDF of the survival times T_j's since observed times are known for all training data whereas survival times are only known for uncensored training data (although the censored data provide information on survival times as well since censored times are lower bounds on survival times).

Lastly, the term $W_3(t|u)$ vanishes when the number of neighbors of u with observed times Y_j's exceeding t_{horizon} is sufficiently large.

For now, we assume that good events $\mathcal{E}_\beta(u)$ and $\mathcal{E}_{\text{horizon}}(u)$ hold, along with Assumptions $\mathbf{A}^{\text{technical}}$ and $\mathbf{A}^{\text{survival}}$ before introducing the third and final good event that comes into play in showing that $W_2(t|u) \to 0$. Lemma 10 guarantees that under these conditions, $\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) \neq 0$ (recall that $t \in [0, t_{\text{horizon}}]$). Moreover, using Assumption $\mathbf{A}^{\text{survival}}$(a) and

since $S_{\mathsf{T}|\mathsf{U}}(\cdot|u)$ monotonically decreases, we have $S_{\mathsf{T}|\mathsf{U}}(t|u) \geq S_{\mathsf{T}|\mathsf{U}}(t_{\text{horizon}}|u) \geq \theta > 0$. Using the fact that for any $a, b \in (0, 1]$, we have $|a - b| \leq |\log a - \log b|$, we get

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad \leq \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\big[(\log \widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad = \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\big[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u) + W_2(t|u) + W_3(t|u))^2\big].
\end{aligned}$$

Next, recall that for any $a_1, a_2, \ldots, a_\ell \in \mathbb{R}$, a consequence of Jensen's inequality is that $(\sum_{i=1}^{\ell} a_i)^2 \leq \ell \sum_{i=1}^{\ell} a_i^2$. By applying this inequality to the RHS above, we get

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad \leq \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\big[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u) + W_2(t|u) + W_3(t|u))^2\big] \\
&\quad \leq 3\underbrace{\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2]}_{\text{term I}} + 3\underbrace{\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_2(t|u))^2]}_{\text{term II}} \\
&\qquad + 3\underbrace{\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_3(t|u))^2]}_{\text{term III}}.
\end{aligned} \tag{A.9}$$

We proceed to bound each of the RHS expectations.

Bounding term I As stated previously, $W_1(t|u)$ is like a kernel regression estimate. By a standard bias-variance decomposition,

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2] \\
&\quad = \underbrace{\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_1(t|u) - \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[W_1(t|u)])^2]}_{\text{variance}} \\
&\qquad + \underbrace{(\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[W_1(t|u)] - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2}_{\text{bias}}.
\end{aligned}$$

We bound the right-hand side.

Lemma 11 (Bound on term I) *Under the same setting as Lemma 10, except where we only condition on good event $\mathcal{E}_\beta(u)$ (i.e., good event $\mathcal{E}_{horizon}(u)$ can optionally hold),*

$$\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2] \leq \underbrace{\frac{K(0)}{4\theta^2 K(\tau)\Xi_u}}_{\textit{variance term bound}} + \underbrace{\frac{(1+\beta)^{2\alpha}\tau^{2\alpha}}{\theta^2}\Big(\lambda_{\mathsf{T}} t + \frac{f_{\mathsf{T}}^* \lambda_{\mathsf{C}} t^2}{2}\Big)^2}_{\textit{bias term bound}}.$$

Proof (Variance term bound) The variance term equals

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_1(t|u) - \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[W_1(t|u)])^2] \\
&= \sum_{i \in \mathcal{N}_Q(u)} \Big(\frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big)^2 \underbrace{\mathbb{E}_{Y_i, D_i|U_i}\Big[\Big(-\frac{D_i \mathbb{1}\{Y_i \leq t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} - \mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}\Big[-\frac{D_i \mathbb{1}\{Y_i \leq t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big]\Big)^2\Big]}_{\text{variance of } -\frac{D_i \mathbb{1}\{Y_i \leq t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} \text{ conditioned on } U_i}.
\end{aligned}$$

Note that this sum is not vacuous since under event $\mathcal{E}_\beta(u)$ and using Lemma 7, $|\mathcal{N}_Q(u)| > 0$. Next, note that $-\frac{D_i \mathbb{1}\{Y_i \le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}$ is a bounded random variable. It is nonzero only when $Y_i \le t$, for which the denominator could be as large as 1 and as small as $S_{\mathsf{Y}|\mathsf{U}}(Y_i|u) \ge S_{\mathsf{Y}|\mathsf{U}}(t|u) \ge S_{\mathsf{Y}|\mathsf{U}}(t_{\text{horizon}}|u) \ge \theta$ using the fact that $S_{\mathsf{Y}|\mathsf{U}}(\cdot|u)$ monotonically decreases and using Assumption $\mathbf{A}^{\text{survival}}$(a). Consequently, $-\frac{D_i \mathbb{1}\{Y_i \le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} \in [-\frac{1}{\theta}, 0]$, which by a standard result on bounded random variables implies that the variance of this random variable is at most $\frac{1}{4\theta^2}$. Hence,

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_1(t|u) - \mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[W_1(t|u)])^2] \le \sum_{i \in \mathcal{N}_Q(u)} \Big(\frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big)^2 \frac{1}{4\theta^2}.$$

Then by Hölder's inequality and Lemma 7,

$$\begin{aligned}
\sum_{i \in \mathcal{N}_Q(u)} \Big(\frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big)^2 \frac{1}{4\theta^2} &\le \frac{1}{4\theta^2}\Big[\max_{i \in \mathcal{N}_Q(u)} \frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big] \underbrace{\sum_{i \in \mathcal{N}_Q(u)} \frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}}_{=1} \\
&\le \frac{1}{4\theta^2} \max_{i \in \mathcal{N}_Q(u)} \frac{K(0)}{K(\tau)|\mathcal{N}_Q(u)|} \\
&= \frac{K(0)}{4\theta^2 K(\tau)|\mathcal{N}_Q(u)|} \le \frac{K(0)}{4\theta^2 K(\tau)\Xi_u},
\end{aligned}$$

which establishes the variance term bound.

(Bias term bound) First, recall from equation (2.1) that the hazard function is $-\frac{\partial}{\partial t} \log S_{\mathsf{T}|\mathsf{U}}(t|u) = \frac{f_{\mathsf{T}|\mathsf{U}}(s|u)}{S_{\mathsf{T}|\mathsf{U}}(s|u)}$. Then by the fundamental theorem of calculus,

$$\begin{aligned}
\log S_{\mathsf{T}|\mathsf{U}}(t|u) &= -\int_0^t \frac{f_{\mathsf{T}|\mathsf{U}}(s|u)}{S_{\mathsf{T}|\mathsf{U}}(s|u)} ds \\
&= -\int_0^t \frac{1}{S_{\mathsf{T}|\mathsf{U}}(s|u) S_{\mathsf{C}|\mathsf{U}}(s|u)} S_{\mathsf{C}|\mathsf{U}}(s|u) f_{\mathsf{T}|\mathsf{U}}(s|u) ds \\
&= -\int_0^t \frac{1}{S_{\mathsf{Y}|\mathsf{U}}(s|u)} S_{\mathsf{C}|\mathsf{U}}(s|u) f_{\mathsf{T}|\mathsf{U}}(s|u) ds, \qquad \text{(A.10)}
\end{aligned}$$

where the last step uses the result that for two independent random variables A_1 and A_2, the random variable $\min\{A_1, A_2\}$ has a tail function (1 minus CDF) that is the product of the tail functions of A_1 and A_2; in our setting, $Y_j = \min\{T_j, C_j\}$ where T_j and C_j are conditionally independent given U_j, so $S_{\mathsf{Y}|\mathsf{U}}(s|u) = S_{\mathsf{T}|\mathsf{U}}(s|u) S_{\mathsf{C}|\mathsf{U}}(s|u)$.

Next, we have

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[W_1(t|u)] = \sum_{i \in \mathcal{N}_Q(u)} \Big(\frac{K_i}{\sum_{j \in \mathcal{N}_Q(u)} K_j}\Big) \mathbb{E}_{Y_i,D_i|U_i}\Big[-\frac{D_i \mathbb{1}\{Y_i \le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big], \qquad \text{(A.11)}$$

where

$$\mathbb{E}_{Y_i,D_i|U_i}\Big[-\frac{D_i\mathbb{1}\{Y_i\le t\}}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big]=-\int_0^t\Big[\int_s^\infty\frac{1}{S_{\mathsf{Y}|\mathsf{U}}(s|u)}d\mathbb{P}_{\mathsf{C}|\mathsf{U}}(c|U_i)\Big]d\mathbb{P}_{\mathsf{T}|\mathsf{U}}(s|U_i)$$
$$=-\int_0^t\frac{1}{S_{\mathsf{Y}|\mathsf{U}}(s|u)}S_{\mathsf{C}|\mathsf{U}}(s|U_i)f_{\mathsf{T}|\mathsf{U}}(s|U_i)ds.$$

We take the difference of equations (A.11) and (A.10) to get

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[W_1(t|u)]-\log S_{\mathsf{T}|\mathsf{U}}(t|u)$$
$$=\sum_{i\in\mathcal{N}_Q(u)}\Big(\frac{K_i}{\sum_{j\in\mathcal{N}_Q(u)}K_j}\Big)\int_0^t\frac{S_{\mathsf{C}|\mathsf{U}}(s|u)f_{\mathsf{T}|\mathsf{U}}(s|u)-S_{\mathsf{C}|\mathsf{U}}(s|U_i)f_{\mathsf{T}|\mathsf{U}}(s|U_i)}{S_{\mathsf{Y}|\mathsf{U}}(s|u)}ds.$$

Note that Assumption $\mathbf{A}^{\text{survival}}$(b) implies that $S_{\mathsf{C}|\mathsf{U}}(s|\cdot)f_{\mathsf{T}|\mathsf{U}}(s|\cdot)$ is Hölder continuous with parameters $(\lambda_{\mathsf{T}}+f_{\mathsf{T}}^*\lambda_{\mathsf{C}}s)$ and α, where $f_{\mathsf{T}}^*=\max_{s\in[0,t_{\text{horizon}}],u\in\mathcal{U}}f_{\mathsf{T}|\mathsf{U}}(s|u)$; this maximum exists by compactness of $[0,t_{\text{horizon}}]$ and $\mathcal{U}$. Then by Hölder's inequality, Assumption $\mathbf{A}^{\text{survival}}$, and Lemma 9,

$$|\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[W_1(t|u)]-\log S_{\mathsf{T}|\mathsf{U}}(t|u)|$$
$$\le\max_{i\in\mathcal{N}_Q(u)}\Big|\int_0^t\frac{1}{\underbrace{S_{\mathsf{Y}|\mathsf{U}}(s|u)}_{\ge S_{\mathsf{Y}|\mathsf{U}}(t_{\text{horizon}}|u)\ge\theta}}[S_{\mathsf{C}|\mathsf{U}}(s|u)f_{\mathsf{T}|\mathsf{U}}(s|u)-S_{\mathsf{C}|\mathsf{U}}(s|U_i)f_{\mathsf{T}|\mathsf{U}}(s|U_i)]ds\Big|$$
$$\le\frac{1}{\theta}\max_{i\in\mathcal{N}_Q(u)}\int_0^t|S_{\mathsf{C}|\mathsf{U}}(s|u)f_{\mathsf{T}|\mathsf{U}}(s|u)-S_{\mathsf{C}|\mathsf{U}}(s|U_i)f_{\mathsf{T}|\mathsf{U}}(s|U_i)|ds$$
$$\le\frac{1}{\theta}\max_{i\in\mathcal{N}_Q(u)}\int_0^t(\lambda_{\mathsf{T}}+f_{\mathsf{T}}^*\lambda_{\mathsf{C}}s)\|u-U_i\|_2^\alpha ds$$
$$\le\frac{1}{\theta}\max_{i\in\mathcal{N}_Q(u)}\int_0^t(\lambda_{\mathsf{T}}+f_{\mathsf{T}}^*\lambda_{\mathsf{C}}s)((1+\beta)\tau)^\alpha ds$$
$$=\frac{(1+\beta)^\alpha\tau^\alpha}{\theta}\int_0^t(\lambda_{\mathsf{T}}+f_{\mathsf{T}}^*\lambda_{\mathsf{C}}s)ds$$
$$=\frac{(1+\beta)^\alpha\tau^\alpha}{\theta}\Big(\lambda_{\mathsf{T}}t+\frac{f_{\mathsf{T}}^*\lambda_{\mathsf{C}}t^2}{2}\Big).$$

Squaring both sides yields the bias term bound. ■

Bounding term II The term $W_2(t|u)$ is related to a CDF estimation problem. Specifically, define the following estimate of $S_{\mathsf{Y}|\mathsf{U}}(s|u)$:

$$\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u):=\frac{d_K^+(u,s)}{\sum_{\ell\in\mathcal{N}_Q(u)}K_\ell}=\sum_{j\in\mathcal{N}_Q(u)}\frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)}K_\ell}\mathbb{1}\{Y_j>s\}.$$

Then $1 - \widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)$ is a weighted empirical distribution that estimates $1 - S_{\mathsf{Y}|\mathsf{U}}(s|u)$. We introduce the third and final good event

$$\mathcal{E}_{\text{CDF}}(u) := \Big\{ \underbrace{\sup_{s\geq 0} |\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - \mathbb{E}_{Y_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)]|}_{\spadesuit :=} \leq \varepsilon_{\text{CDF}} \Big\}, \tag{A.12}$$

where

$$\varepsilon_{\text{CDF}} = \sqrt{\frac{9K^2(0)}{4K^2(\tau)|\mathcal{N}_Q(u)|} \log\Big(|\mathcal{N}_Q(u)|^3 \frac{144K^2(\tau)}{9K^2(0)}\Big)}.$$

We show later (Lemma 14) that this event holds with high probability for large n. Note that event $\mathcal{E}_{\text{CDF}}(u)$ is like a variance term, saying that $\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)$ is close to its expectation. We also define a bias term

$$\heartsuit := \sup_{s\in[0,t]} |\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - \mathbb{E}_{Y_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)]|. \tag{A.13}$$

Then the following lemma relates $W_2(t|u)$ to the variance and bias terms.

Lemma 12 (Bound on term II) *Under the same setting as Lemma 10, except where we now condition on all three good events $\mathcal{E}_\beta(u)$, $\mathcal{E}_{horizon}(u)$, and $\mathcal{E}_{CDF}(u)$ holding,*

$$(W_2(t|u))^2 \leq \frac{12K^2(0)}{K^2(\tau)\theta^4\Xi_u^2} + \frac{12K^2(0)}{K^2(\tau)\theta^4}\Big(\spadesuit^2 + \heartsuit^2\Big).$$

where

$$\spadesuit^2 \leq \varepsilon_{CDF}^2, \quad \heartsuit^2 \leq (\lambda_{\mathsf{T}} + \lambda_{\mathsf{C}})^2 t^2 (1+\beta)^{2\alpha}\tau^{2\alpha}.$$

Thus,

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_2(t|u))^2] \leq \frac{12K^2(0)}{K^2(\tau)\theta^4\Xi_u^2} + \frac{12K^2(0)}{K^2(\tau)\theta^4}[\varepsilon_{CDF}^2 + (\lambda_{\mathsf{T}} + \lambda_{\mathsf{C}})^2 t^2 (1+\beta)^{2\alpha}\tau^{2\alpha}].$$

Proof We use Hölder's inequality and a bit of algebra to get

$$\begin{aligned}
|W_2(t|u)| &= \Big| \sum_{i\in\mathcal{N}_Q(u)} \frac{K_i D_i \mathbb{1}\{Y_i \leq t\}\big[\frac{1}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} - \frac{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell}{K_i + d_K^+(u,Y_i)}\big]}{\sum_{j\in\mathcal{N}_Q(u)} K_j} \Big| \\
&\leq \max_{i\in\mathcal{N}_Q(u)} \Big| D_i \mathbb{1}\{Y_i \leq t\}\Big[\frac{1}{S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} - \frac{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell}{K_i + d_K^+(u,Y_i)}\Big]\Big| \\
&= \max_{i\in\mathcal{N}_Q(u)} \Big| \frac{D_i \mathbb{1}\{Y_i \leq t\}\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell}{(K_i + d_K^+(u,Y_i))S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big[\frac{K_i + d_K^+(u,Y_i)}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} - S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)\Big]\Big| \\
&= \max_{i\in\mathcal{N}_Q(u)} \Big| \underbrace{\frac{D_i \mathbb{1}\{Y_i \leq t\}\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell}{(K_i + d_K^+(u,Y_i))S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}}_{\clubsuit_i :=}\Big[\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} + \underbrace{\widehat{S}_{\mathsf{Y}|\mathsf{U}}(Y_i|u) - S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}_{\text{CDF/tail estimation}}\Big]\Big| \\
&\leq \sup_{\substack{i\in\mathcal{N}_Q(u),\\ s\in[0,t]}} \Big|\clubsuit_i\Big[\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} + \widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - S_{\mathsf{Y}|\mathsf{U}}(s|u)\Big]\Big|,
\end{aligned} \tag{A.14}$$

where the last step uses the fact that for $\clubsuit_i$ to be nonzero, we must have $Y_i \le t$, for which we then replace Y_i with a worst case $s \in [0,t]$ in the CDF/tail estimation problem.

By squaring both sides of inequality (A.14) (and noting that the square can go into the sup on the RHS) followed by using the fact that $(\sum_{i=1}^{\ell} a_i)^2 \le \ell \sum_{i=1}^{\ell} a_i^2$,

$$
\begin{aligned}
(W_2(t|u))^2 &\le \sup_{\substack{i\in\mathcal{N}_Q(u),\\ s\in[0,t]}} \clubsuit_i^2 \Big[\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} + \widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - S_{\mathsf{Y}|\mathsf{U}}(s|u) \Big]^2 \\
&= \sup_{\substack{i\in\mathcal{N}_Q(u),\\ s\in[0,t]}} \clubsuit_i^2 \Big[\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} + \widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - \mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)] \\
&\qquad\qquad + \mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)] - S_{\mathsf{Y}|\mathsf{U}}(s|u) \Big]^2 \\
&\le \sup_{\substack{i\in\mathcal{N}_Q(u),\\ s\in[0,t]}} \Big[3\clubsuit_i^2 \Big(\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} \Big)^2 + 3\clubsuit_i^2 (\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - \mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)])^2 \\
&\qquad\qquad + 3\clubsuit_i^2 (\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)] - S_{\mathsf{Y}|\mathsf{U}}(s|u))^2 \Big] \\
&\le 3 \max_{i\in\mathcal{N}_Q(u)} \clubsuit_i^2 \Big(\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} \Big)^2 + \Big(3 \max_{i\in\mathcal{N}_Q(u)} \clubsuit_i^2 \Big) \spadesuit^2 + \Big(3 \max_{i\in\mathcal{N}_Q(u)} \clubsuit_i^2 \Big) \heartsuit^2,
\end{aligned}
\tag{A.15}
$$

where the variance term $\spadesuit$ and bias term $\heartsuit$ are defined in equations (A.12) and (A.13). We now bound $\clubsuit_i^2 (\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell})^2$, $\clubsuit_i^2$, and $\heartsuit^2$.

Since $d_K^+(u,\cdot)$ and $S_{\mathsf{Y}|\mathsf{U}}(\cdot|u)$ are monotonically decreasing and with the constraint in the numerator that $Y_i \le t$, we have

$$
\begin{aligned}
\clubsuit_i^2 \Big(\frac{K_i}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} \Big)^2 &= \Big(\frac{D_i \mathbb{1}\{Y_i \le t\} K_i}{(K_i + d_K^+(u,Y_i)) S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)} \Big)^2 \\
&\le \Big(\frac{D_i \mathbb{1}\{Y_i \le t\} K_i}{d_K^+(u,t) S_{\mathsf{Y}|\mathsf{U}}(t|u)} \Big)^2 \\
&\le \Big(\frac{D_i \mathbb{1}\{Y_i \le t\} K_i}{K(\tau) \cdot \frac{|\mathcal{N}_Q(u)|\theta}{2} \cdot \theta} \Big)^2 && \text{by } \mathcal{E}_{\text{horizon}}(u), \mathbf{A}^{\text{survival}}(\text{a}) \\
&\le \Big(\frac{K(0)}{K(\tau) \cdot \frac{|\mathcal{N}_Q(u)|\theta}{2} \cdot \theta} \Big)^2 \\
&= \frac{4K^2(0)}{K^2(\tau)\theta^4 |\mathcal{N}_Q(u)|^2} \\
&\le \frac{4K^2(0)}{K^2(\tau)\theta^4 \Xi_u^2} && \text{by Lemma 7.}
\end{aligned}
\tag{A.16}
$$

By similar ideas,

$$\clubsuit_i^2 = \Big(\frac{D_i \mathbb{1}\{Y_i \le t\} \sum_{\ell \in \mathcal{N}_Q(u)} K_\ell}{(K_i + d_K^+(u, Y_i)) S_{\mathsf{Y}|\mathsf{U}}(Y_i|u)}\Big)^2 \le \Big(\frac{|\mathcal{N}_Q(u)| K(0)}{(K(\tau)|\mathcal{N}_Q(u)|\theta/2)\theta}\Big)^2 = \frac{4K^2(0)}{K^2(\tau)\theta^4}. \tag{A.17}$$

Lastly, we bound $\heartsuit = \sup_{s\in[0,t]} \diamondsuit(s)$, where $\diamondsuit(s) := |\mathbb{E}_{Y_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)] - S_{\mathsf{Y}|\mathsf{U}}(s|u)|$. For $s \in [0, t]$, we have

$$\begin{aligned}
\mathbb{E}_{Y_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)] &= \mathbb{E}_{Y_{1:n}|U_{1:n}}\Bigg[\sum_{j\in\mathcal{N}_Q(u)} \frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} \mathbb{1}\{Y_j > s\}\Bigg] \\
&= \sum_{j\in\mathcal{N}_Q(u)} \frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} \mathbb{E}_{Y_j|U_j}[\mathbb{1}\{Y_j > s\}]. \\
&= \sum_{j\in\mathcal{N}_Q(u)} \frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} S_{\mathsf{Y}|\mathsf{U}}(s|U_j).
\end{aligned} \tag{A.18}$$

Note that Assumption $\mathbf{A}^{\text{survival}}$(b) implies that $S_{\mathsf{Y}|\mathsf{U}}(s|\cdot)$ is Hölder continuous with parameters $(\lambda_\mathsf{T} + \lambda_\mathsf{C})s$ and α. Then using equation (A.18), Hölder's inequality, Hölder continuity, and Lemma 9,

$$\begin{aligned}
\diamondsuit(s) &= \Bigg| \sum_{j\in\mathcal{N}_Q(u)} \frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell} (S_{\mathsf{Y}|\mathsf{U}}(s|U_j) - S_{\mathsf{Y}|\mathsf{U}}(s|u)) \Bigg| \\
&\le \max_{j\in\mathcal{N}_Q(u)} |S_{\mathsf{Y}|\mathsf{U}}(s|U_j) - S_{\mathsf{Y}|\mathsf{U}}(s|u)| \\
&\le \max_{j\in\mathcal{N}_Q(u)} (\lambda_\mathsf{T} + \lambda_\mathsf{C}) s \|U_j - u\|_2^\alpha \\
&\le \max_{j\in\mathcal{N}_Q(u)} (\lambda_\mathsf{T} + \lambda_\mathsf{C}) s (1+\beta)^\alpha \tau^\alpha \\
&= (\lambda_\mathsf{T} + \lambda_\mathsf{C}) t (1+\beta)^\alpha \tau^\alpha.
\end{aligned}$$

Therefore,

$$\heartsuit = \sup_{s\in[0,t]} \diamondsuit(s) \le (\lambda_\mathsf{T} + \lambda_\mathsf{C}) t (1+\beta)^\alpha \tau^\alpha. \tag{A.19}$$

Combining inequalities (A.15), (A.16), (A.17), and (A.19) yields the claim. ■

Bounding term III Lastly, we have the following bound.

Lemma 13 (Bound on term III) *Under the same setting as Lemma 10 (conditioning on $\mathcal{E}_\beta(u)$ and $\mathcal{E}_{horizon}(u)$) and with the addition of Assumption $\mathbf{A}^{intrinsic}$ and the requirement that $n \ge \frac{2K(0)}{K(\tau)\theta C_{d'}((1-\beta)\tau)^{d'}}$, we have*

$$\mathbb{E}_{Y_{1:n}, D_{1:n}|U_{1:n}}[(W_3(t|u))^2] \le \frac{16K^4(0)}{K^4(\tau)\theta^4\Xi_u^2}.$$

Proof We write $|W_3(t|u)| = \sum_{i\in\mathcal{N}_Q(u)} \Upsilon_i$, where

$$\Upsilon_i := D_i \mathbb{1}\{Y_i \le t\} \sum_{\ell=2}^{\infty} \frac{1}{\ell[1+(d_K^+(u,Y_i)/K_i)]^\ell}.$$

Using the constraint that $Y_i \le t$, that $d_K^+(u,\cdot)$ monotonically decreases, and that $t \le t_{\text{horizon}}$, we have

$$\begin{aligned}\Upsilon_i &\le D_i \mathbb{1}\{Y_i \le t\} \sum_{\ell=2}^{\infty} \frac{1}{\ell[1+(d_K^+(u,t)/K_i)]^\ell} \\ &\le \sum_{\ell=2}^{\infty} \frac{1}{\ell[1+(d_K^+(u,t)/K_i)]^\ell} \\ &\le \sum_{\ell=2}^{\infty} \frac{1}{\ell[1+(d_K^+(u,t_{\text{horizon}})/K_i)]^\ell}.\end{aligned}$$

Next, recall that good event $\mathcal{E}_{\text{horizon}}(u)$ ensures that $d^+(u,t_{\text{horizon}}) > \frac{|\mathcal{N}_Q(u)|\theta}{2}$. Furthermore, $d_K^+(u,t_{\text{horizon}}) \ge K(\tau) d^+(u,t_{\text{horizon}})$, and $K_i \le K(0)$, so

$$\frac{1}{\ell[1+(d_K^+(u,t_{\text{horizon}})/K_i)]^\ell} \le \frac{1}{\ell\big[1+\frac{K(\tau)\frac{|\mathcal{N}_Q(u)|\theta}{2}}{K(0)}\big]^\ell}.$$

Hence,

$$\Upsilon_i \le \sum_{\ell=2}^{\infty} \frac{1}{\ell\big[1+\frac{K(\tau)|\mathcal{N}_Q(u)|\theta}{2K(0)}\big]^\ell}.$$

Noting that $\sum_{\ell=2}^{\infty} \frac{1}{\ell(1+z)^\ell} = \log(1+\frac{1}{z}) - \frac{1}{1+z} \le \frac{1}{(1+z)^2}$ for all $z \ge 0.46241$, then provided that

$$\frac{K(\tau)|\mathcal{N}_Q(u)|\theta}{2K(0)} \ge 0.46241, \tag{A.20}$$

we have

$$\Upsilon_i \le \frac{1}{(1+\frac{K(\tau)|\mathcal{N}_Q(u)|\theta}{2K(0)})^2} \le \frac{1}{(\frac{K(\tau)|\mathcal{N}_Q(u)|\theta}{2K(0)})^2} = \frac{4K^2(0)}{K^2(\tau)|\mathcal{N}_Q(u)|^2\theta^2}.$$

Thus,

$$\begin{aligned}|W_3(t|u)| &= \sum_{i\in\mathcal{N}_Q(u)} \Upsilon_i \le \sum_{i\in\mathcal{N}_Q(u)} \frac{4K^2(0)}{K^2(\tau)|\mathcal{N}_Q(u)|^2\theta^2} \\ &= \frac{4K^2(0)}{K^2(\tau)|\mathcal{N}_Q(u)|\theta^2} \le \frac{4K^2(0)}{K^2(\tau)\theta^2} \cdot \frac{1}{\Xi_u}.\end{aligned}$$

Squaring and taking expectation $\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}$ of both sides, we have

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_3(t|u))^2] \le \frac{16K^4(0)}{K^4(\tau)\theta^4} \cdot \frac{1}{\Xi_u^2}.$$

The only missing piece is to ensure that condition (A.20) holds. Under event $\mathcal{E}_\beta(u)$ and Assumption $\mathbf{A}^{\text{intrinsic}}$, with the help of Lemma 7,

$$|\mathcal{N}_Q(u)| \geq \Xi_u > \frac{1}{2} n \mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau)) \geq \frac{1}{2} n C_{d'}((1-\beta)\tau)^{d'}.$$

Thus, since we assume that $n \geq \frac{2K(0)}{K(\tau)\theta C_{d'}((1-\beta)\tau)^{d'}}$, we have

$$\begin{aligned}
\frac{K(\tau)|\mathcal{N}_Q(u)|\theta}{2} &\geq \frac{K(\tau)\frac{1}{2} n C_{d'}((1-\beta)\tau)^{d'}\theta}{2} \\
&\geq \frac{K(\tau)\frac{1}{2}[\frac{2K(0)}{K(\tau)\theta C_{d'}((1-\beta)\tau)^{d'}}] C_{d'}((1-\beta)\tau)^{d'}\theta}{2} \\
&= \frac{1}{2} \geq 0.46241,
\end{aligned}$$

which verifies that condition (A.20) holds. ■

Deriving a final pointwise bound Putting together bound (A.9) and Lemmas 11, 12, and 13, we have that when all three good events $\mathcal{E}_\beta(u)$, $\mathcal{E}_{\text{horizon}}(u)$, and $\mathcal{E}_{\text{CDF}}(u)$ hold, and $n \geq \frac{2K(0)}{K(\tau)\theta C_{d'}((1-\beta)\tau)^{d'}}$,

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad\leq 3\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_1(t|u) - \log S_{\mathsf{T}|\mathsf{U}}(t|u))^2] + 3\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_2(t|u))^2] \\
&\quad+ 3\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}[(W_3(t|u))^2] \\
&\quad\leq 3\left[\frac{K(0)}{4\theta^2 K(\tau)\Xi_u} + \frac{(1+\beta)^{2\alpha}\tau^{2\alpha}}{\theta^2}\Big(\lambda_{\mathsf{T}} t + \frac{f_{\mathsf{T}}^* \lambda_{\mathsf{C}} t^2}{2}\Big)^2\right] \\
&\qquad+ 3\left[\frac{12K^2(0)}{K^2(\tau)\Xi_u^2\theta^4} + \frac{12K^2(0)}{K^2(\tau)\theta^4}\varepsilon_{\text{CDF}}^2 + \frac{12K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2 t^2 (1+\beta)^{2\alpha}\tau^{2\alpha}\right] \\
&\qquad+ 3\left[\frac{16K^4(0)}{K^4(\tau)\theta^4\Xi_u^2}\right] \\
&\quad\leq \frac{1}{n}\cdot\frac{332K^4(0)}{\theta^4 K^4(\tau)\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}\log\Big(n^3\frac{144K^2(\tau)}{9K^2(0)}\Big) \\
&\qquad+ (1+\beta)^{2\alpha}\tau^{2\alpha}\left[\frac{3}{\theta^2}\Big(\lambda_{\mathsf{T}} t + \frac{f_{\mathsf{T}}^*\lambda_{\mathsf{C}} t^2}{2}\Big)^2 + \frac{36K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2 t^2\right].
\end{aligned}$$

When not all good events occur, we resort to the trivial bound

$$\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \leq 1.$$

Abbreviating the three good events as $\mathcal{E}_{\text{good}}(u) := \mathcal{E}_\beta(u) \cup \mathcal{E}_{\text{horizon}}(u) \cup \mathcal{E}_{\text{CDF}}(u)$,

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n},D_{1:n},U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad= \mathbb{E}_{U_{1:n}}\big[\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\mathbb{1}\{\mathcal{E}_{\text{good}}(u)\} \\
&\qquad\qquad + (\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\mathbb{1}\{\mathcal{E}^c_{\text{good}}(u)\}\big]\big] \\
&\quad\le \mathbb{E}_{U_{1:n}}\big[\mathbb{E}_{Y_{1:n},D_{1:n}|U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\mathbb{1}\{\mathcal{E}_{\text{good}}(u)\} + \mathbb{1}\{\mathcal{E}^c_{\text{good}}(u)\}\big]\big] \\
&\quad\le \frac{1}{n}\cdot\frac{332K^4(0)}{\theta^4K^4(\tau)\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}\log\Big(n^3\frac{144K^2(\tau)}{9K^2(0)}\Big) \\
&\qquad + (1+\beta)^{2\alpha}\tau^{2\alpha}\left[\frac{3}{\theta^2}\Big(\lambda_{\mathsf{T}}t + \frac{f^*_{\mathsf{T}}\lambda_{\mathsf{C}}t^2}{2}\Big)^2 + \frac{36K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2t^2\right] \\
&\qquad + \mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}^c_{\text{good}}(u)\}].
\end{aligned} \tag{A.21}$$

We next bound $\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}^c_{\text{good}}(u)\}]$.

Lemma 14 (The survival analysis good events hold with high probability) *Consider the same setting as Lemma 10 except without conditioning on any good events. Moreover, we add Assumption* $\mathbf{A}^{intrinsic}$ *and require that* $n \ge \frac{2}{C_{d'}((1-\beta)\tau)^{d'}}\big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\big)^{1/3}$. *Then*

$$\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}^c_{good}(u)\}] \le \frac{14}{\theta^2 n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}.$$

We defer the proof of this lemma to Appendix A.3.2 as it is somewhat technical.

Putting together bound (A.21) and Lemma 14, and in particular absorbing Lemma 14's bound into the leading error term in (A.21), we get the final pointwise bound: for $n \ge \frac{2}{C_{d'}((1-\beta)\tau)^{d'}}\max\big\{\frac{K(0)}{K(\tau)\theta}, \big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\big)^{1/3}\big\}$,

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n},D_{1:n},U_{1:n}}\big[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|u) - S_{\mathsf{T}|\mathsf{U}}(t|u))^2\big] \\
&\quad\le \frac{346K^4(0)}{n\theta^4K^4(\tau)\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}\log\Big(n^3\frac{144K^2(\tau)}{9K^2(0)}\Big) \\
&\qquad + (1+\beta)^{2\alpha}\tau^{2\alpha}\left[\frac{3}{\theta^2}\Big(\lambda_{\mathsf{T}}t + \frac{f^*_{\mathsf{T}}\lambda_{\mathsf{C}}t^2}{2}\Big)^2 + \frac{36K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2t^2\right].
\end{aligned} \tag{A.22}$$

Completing the proof Finally, we account for randomness in the test embedding vector $U \sim \mathbb{P}_{\mathsf{U}}$ and integrate over time. Suppose that $n \ge \frac{2}{C_{d'}((1-\beta)\tau)^{d'}}\max\big\{\frac{K(0)}{K(\tau)\theta}, \big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\big)^{1/3}\big\}$. By Fubini's theorem, iterated expecta-

tion, and the final pointwise bound (A.22),

$$\begin{aligned}
&\mathbb{E}_{Y_{1:n},D_{1:n},U_{1:n},U}\Big[\frac{\int_0^{t_{\text{horizon}}}(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|U)-S_{\mathsf{T}|\mathsf{U}}(t|U))^2dt}{t_{\text{horizon}}}\Big]\\
&\quad= \int_{[0,t_{\text{horizon}}]}\frac{\mathbb{E}_U[\mathbb{E}_{Y_{1:n},D_{1:n},U_{1:n}|U}[(\widehat{S}_{\mathsf{T}|\mathsf{U}}(t|U)-S_{\mathsf{T}|\mathsf{U}}(t|U))^2]]}{t_{\text{horizon}}}dt\\
&\quad\le \int_{[0,t_{\text{horizon}}]}\frac{1}{t_{\text{horizon}}}\mathbb{E}_U\Bigg[\frac{1}{n}\cdot\frac{346K^4(0)\log(n^3\frac{144K^2(\tau)}{9K^2(0)})}{\theta^4K^4(\tau)\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}\\
&\qquad\qquad+(1+\beta)^{2\alpha}\tau^{2\alpha}\Bigg[\frac{3}{\theta^2}\Big(\lambda_{\mathsf{T}}t+\frac{f^*_{\mathsf{T}}\lambda_{\mathsf{C}}t^2}{2}\Big)^2+\frac{36K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2t^2\Bigg]\Bigg]dt\\
&\quad=\frac{346K^4(0)\log(n^3\frac{144K^2(\tau)}{9K^2(0)})}{n\theta^4K^4(\tau)}\mathbb{E}_U\Big[\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}\Big]\\
&\qquad+(1+\beta)^{2\alpha}\tau^{2\alpha}\Bigg[\frac{t^2_{\text{horizon}}}{\theta^2}\Big(\lambda^2_{\mathsf{T}}+\frac{3\lambda_{\mathsf{T}}f^*_{\mathsf{T}}\lambda_{\mathsf{C}}t_{\text{horizon}}}{4}+\frac{3(f^*_{\mathsf{T}})^2\lambda^2_{\mathsf{C}}t^2_{\text{horizon}}}{20}\Big)\\
&\qquad\qquad+\frac{12K^2(0)}{K^2(\tau)\theta^4}(\lambda_{\mathsf{T}}+\lambda_{\mathsf{C}})^2t^2_{\text{horizon}}\Bigg].
\end{aligned}$$

At this point, we can bound $\mathbb{E}_U[\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}]$. This is where covering numbers and intrinsic dimension come into play.

Case 1 (using a covering argument) Let $\zeta_1,\zeta_2,\ldots,\zeta_N$ be a $\frac{(1-\beta)\tau}{2}$-cover of $\mathcal{U}$ that has the smallest size possible, i.e., $N=N(\frac{(1-\beta)\tau}{2};\mathcal{U})$. Then

$$\mathbb{E}_U\Big[\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}\Big]\le\sum_{j=1}^N\mathbb{E}_U\Big[\frac{\mathbb{1}\{U\in B(\zeta_j,\frac{(1-\beta)\tau}{2})\}}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}\Big]. \tag{A.23}$$

Note that $U\in B(\zeta_j,\frac{(1-\beta)\tau}{2})$ implies that $B(U,(1-\beta)\tau)$ contains $B(\zeta_j,\frac{(1-\beta)\tau}{2})$. To see this, let u' be any point in $B(\zeta_j,\frac{(1-\beta)\tau}{2})$. Then by the triangle inequality,

$$\|u'-U\|_2\le\underbrace{\|u'-\zeta_j\|_2}_{\le\frac{(1-\beta)\tau}{2}}+\underbrace{\|\zeta_j-U\|_2}_{\frac{(1-\beta)\tau}{2}}\le(1-\beta)\tau,$$

i.e., u' is also in $B(U,(1-\beta)\tau)$. Hence, $U\in B(\zeta_j,\frac{(1-\beta)\tau}{2})$ implies that $B(U,(1-\beta)\tau)\supseteq B(\zeta_j,\frac{(1-\beta)\tau}{2})$, which in turn implies that $\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))\ge\mathbb{P}_{\mathsf{U}}(B(\zeta_j,\frac{(1-\beta)\tau}{2}))$. Thus, the RHS of inequality (A.23) can be bounded as

$$\sum_{j=1}^N\mathbb{E}_U\Big[\frac{\mathbb{1}\{U\in B(\zeta_j,\frac{(1-\beta)\tau}{2})\}}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}\Big]\le\sum_{j=1}^N\mathbb{E}_U\Big[\frac{\mathbb{1}\{U\in B(\zeta_j,\frac{(1-\beta)\tau}{2})\}}{\mathbb{P}_{\mathsf{U}}(B(\zeta_j,\frac{(1-\beta)\tau}{2}))}\Big]=N=N(\tfrac{(1-\beta)\tau}{2};\mathcal{U}),$$

at which point we conclude that $\mathbb{E}_U[\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}]\le N(\frac{(1-\beta)\tau}{2};\mathcal{U})$.

Case 2 (using Assumption $\mathbf{A}^{intrinsic}$) For all $U \in \mathcal{U}$, we have $\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))} \leq \frac{1}{C_{d'}((1-\beta)\tau)^{d'}}$. Hence, $\mathbb{E}_U[\frac{1}{\mathbb{P}_{\mathsf{U}}(B(U,(1-\beta)\tau))}] \leq \frac{1}{C_{d'}((1-\beta)\tau)^{d'}}$. ■

A.3.2 Proof of Lemma 14

Observe that

$$\begin{aligned}\mathcal{E}_{\text{good}}(u) &= (\mathcal{E}_\beta(u) \cap \mathcal{E}_{\text{horizon}}(u) \cap \mathcal{E}_{\text{CDF}}(u))^c \\ &= \mathcal{E}_\beta^c(u) \cup \mathcal{E}_{\text{horizon}}^c(u) \cup \mathcal{E}_{\text{CDF}}^c(u) \\ &= \mathcal{E}_\beta^c(u) \cup [\mathcal{E}_{\text{horizon}}^c(u) \cap \mathcal{E}_\beta(u)] \cup [\mathcal{E}_{\text{CDF}}^c(u) \cap \mathcal{E}_\beta(u)].\end{aligned}$$

Hence, by a union bound

$$\begin{aligned}&\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}_{\text{good}}(u)\}] \\ &\quad\leq \underbrace{\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}_\beta^c(u)\}]}_{(a)} + \underbrace{\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}_{\text{horizon}}^c(u) \cap \mathcal{E}_\beta(u)\}]}_{(b)} \\ &\qquad + \underbrace{\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}_{\text{CDF}}^c(u) \cap \mathcal{E}_\beta(u)\}]}_{(c)}. \end{aligned} \tag{A.24}$$

We upper-bound each of these three terms next.

Term (a) This term is bounded precisely by Lemma 8. In particular,

$$\text{term } (a) \leq \frac{8}{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}. \tag{A.25}$$

Term (b) For term (b), note that within the expectation there is no dependence on $D_{1:n}$. In particular,

$$\begin{aligned}&\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{\mathcal{E}_{\text{horizon}}^c(u) \cap \mathcal{E}_\beta(u)\}] \\ &\quad= \mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \leq \frac{|\mathcal{N}_Q(u)|\theta}{2}\Big\}\mathbb{1}\Big\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\Big\}\Big] \\ &\quad= \mathbb{E}_{U_{1:n},Y_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \leq \frac{|\mathcal{N}_Q(u)|\theta}{2}\Big\}\mathbb{1}\Big\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\Big\}\Big] \\ &\quad= \mathbb{E}_{U_{1:n}}\Big[\mathbb{1}\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\}\mathbb{E}_{Y_{1:n}|U_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \leq \frac{|\mathcal{N}_Q(u)|\theta}{2}\Big\}\Big]\Big]. \end{aligned} \tag{A.26}$$

By Assumption $\mathbf{A}^{\text{survival}}$(a),

$$|\mathcal{N}_Q(u)|\theta = \sum_{j\in\mathcal{N}_Q(u)} \theta \leq \sum_{j\in\mathcal{N}_Q(u)} S_{\mathsf{Y}|\mathsf{U}}(t_{\text{horizon}}|U_j) =: \mu^+. \tag{A.27}$$

Hence, $d^+(u,t_{\text{horizon}}) \leq \frac{|\mathcal{N}_Q(u)|\theta}{2}$ implies that $d^+(u,t_{\text{horizon}}) \leq \frac{\mu^+}{2}$, i.e.,

$$\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \leq \frac{|\mathcal{N}_Q(u)|\theta}{2}\Big\} \leq \mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \leq \frac{\mu^+}{2}\Big\}.$$

Thus,

$$\mathbb{E}_{U_{1:n}}\Big[\mathbb{1}\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\}\mathbb{E}_{Y_{1:n}|U_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \le \frac{|\mathcal{N}_Q(u)|\theta}{2}\Big\}\Big]\Big]$$
$$\le \mathbb{E}_{U_{1:n}}\Big[\mathbb{1}\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\}\mathbb{E}_{Y_{1:n}|U_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \le \frac{\mu^+}{2}\Big\}\Big]\Big]. \quad \text{(A.28)}$$

We now upper-bound $\mathbb{E}_{Y_{1:n}|U_{1:n}}[\mathbb{1}\{d^+(u,t_{\text{horizon}}) \le \frac{\mu^+}{2}\}]$, under the constraint that $\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))$. Importantly, since the $Y_{1:n}$ variables are conditionally independent given $U_{1:n}$ and since $\mathcal{N}_Q(u)$ is deterministic given $U_{1:n}$, then after conditioning on $U_{1:n}$, the sum

$$d^+(u,t_{\text{horizon}}) = \sum_{j\in\mathcal{N}_Q(u)} \mathbb{1}\{Y_j > t_{\text{horizon}}\}$$

is over independent random variables and, furthermore, using Lemma 7, the above summation is not vacuous since $|\mathcal{N}_Q(u)| \ge \Xi_u > 0$. Thus, with $\mathcal{N}_Q(u)$ nonempty, the expectation of the above sum is

$$\mathbb{E}_{Y_{1:n}|U_{1:n}}[d^+(u,t_{\text{horizon}})] = \sum_{j\in\mathcal{N}_Q(u)} \mathbb{E}_{Y_j|U_j}[\mathbb{1}\{Y_j > t_{\text{horizon}}\}] = \sum_{j\in\mathcal{N}_Q(u)} S_{\mathsf{Y}|\mathsf{U}}(t_{\text{horizon}}|U_j) = \mu^+.$$

Applying Hoeffding's inequality

$$\begin{aligned}
\mathbb{E}_{Y_{1:n}|U_{1:n}}\Big[\mathbb{1}\Big\{d^+(u,t_{\text{horizon}}) \le \frac{\mu^+}{2}\Big\}\Big] &\le \exp\Big(-\frac{2(\frac{1}{2}\mu^+)^2}{|\mathcal{N}_Q(u)|}\Big) \\
&\le \exp\Big(-\frac{2(\frac{|\mathcal{N}_Q(u)|\theta}{2})^2}{|\mathcal{N}_Q(u)|}\Big) && \text{by inequality (A.27)} \\
&= \exp\Big(-\frac{\theta^2}{2}|\mathcal{N}_Q(u)|\Big) \\
&\le \exp\Big(-\frac{\theta^2}{2}\Xi_u\Big) && \text{by Lemma 7} \\
&\le \frac{2}{\theta^2\Xi_u} \\
&< \frac{4}{\theta^2 n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))} && \text{by good event } \mathcal{E}_\beta(u),
\end{aligned} \quad \text{(A.29)}$$

where the last step uses the constraint $\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))$.

Stringing together inequalities (A.26), (A.28), and (A.29), we see that

$$\text{term } (b) \le \frac{4}{\theta^2 n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}. \quad \text{(A.30)}$$

Term (c) Recall that we use the shorthand notation $\spadesuit = \sup_{s\geq 0}|\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) - \mathbb{E}_{Y_{1:n}|U_{1:n}}[\widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u)]|$. Then term (c) can be written as

$$\begin{aligned}
&\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}\Big[\mathbb{1}\{\spadesuit > \varepsilon_{\text{CDF}}\}\mathbb{1}\Big\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\Big\}\Big] \\
&\quad= \mathbb{E}_{U_{1:n},Y_{1:n}}\Big[\mathbb{1}\{\spadesuit > \varepsilon_{\text{CDF}}\}\mathbb{1}\Big\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\Big\}\Big] \\
&\quad= \mathbb{E}_{U_{1:n}}\Big[\mathbb{1}\Big\{\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))\Big\}\mathbb{E}_{Y_{1:n}|U_{1:n}}\Big[\mathbb{1}\{\spadesuit > \varepsilon_{\text{CDF}}\}\Big]\Big]. \qquad \text{(A.31)}
\end{aligned}$$

Under the constraint that $\Xi_u > \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))$ and conditioned on $U_{1:n}$,

$$1 - \widehat{S}_{\mathsf{Y}|\mathsf{U}}(s|u) = \sum_{j\in\mathcal{N}_Q(u)} \frac{K_j}{\sum_{\ell\in\mathcal{N}_Q(u)} K_\ell}\mathbb{1}\{Y_j \leq s\}$$

is a weighted empirical distribution constructed from more than $\frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))$ samples. Applying Proposition 3.1 of Chen (2019),

$$\begin{aligned}
\mathbb{E}_{Y_{1:n}|U_{1:n}}[\mathbb{1}\{\spadesuit > \varepsilon_{\text{CDF}}\}] &\leq \frac{6}{\varepsilon_{\text{CDF}}}\exp\Big(-\frac{2\varepsilon_{\text{CDF}}^2(\sum_{j\in\mathcal{N}_Q(u)}K_j)^2}{9\sum_{\ell\in\mathcal{N}_Q(u)}K_\ell^2}\Big) \\
&\leq \frac{6}{\varepsilon_{\text{CDF}}}\exp\Big(-\frac{2\varepsilon_{\text{CDF}}^2(|\mathcal{N}_Q(u)|K(\tau))^2}{9|\mathcal{N}_Q(u)|K^2(0)}\Big) \\
&= \frac{6}{\varepsilon_{\text{CDF}}}\exp\Big(-\frac{2\varepsilon_{\text{CDF}}^2K^2(\tau)}{9K^2(0)}|\mathcal{N}_Q(u)|\Big). \qquad \text{(A.32)}
\end{aligned}$$

We show that our choice of ε_{CDF} ensures that the RHS is at most $\frac{1}{|\mathcal{N}_Q(u)|}$, i.e., we want to show that

$$\frac{6}{\varepsilon_{\text{CDF}}}\exp\Big(-\frac{2\varepsilon_{\text{CDF}}^2K^2(\tau)}{9K^2(0)}|\mathcal{N}_Q(u)|\Big) \leq \frac{1}{|\mathcal{N}_Q(u)|}. \qquad \text{(A.33)}$$

Lemma D.1(b) of Chen (2019) says that inequality (A.33) holds under the sufficient conditions

$$\varepsilon_{\text{CDF}} < 6|\mathcal{N}_Q(u)| \quad \text{and} \quad |\mathcal{N}_Q(u)| \geq \Big(\frac{9K^2(0)e}{144K^2(\tau)}\Big)^{1/3}. \qquad \text{(A.34)}$$

The latter holds since we assume that $n \geq \frac{2}{C_{d'}((1-\beta)\tau)^{d'}}\big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\big)^{1/3}$, so using good event $\mathcal{E}_\beta(u)$, Lemma 7 and Assumption $\mathbf{A}^{\text{intrinsic}}$, we have

$$\begin{aligned}
|\mathcal{N}_Q(u)| &\geq \Xi_u \\
&> \frac{1}{2}n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau)) \\
&\geq \frac{1}{2}nC_{d'}((1-\beta)\tau)^{d'} \\
&\geq \frac{1}{2}\Big[\frac{2}{C_{d'}((1-\beta)\tau)^{d'}}\Big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\Big)^{1/3}\Big]C_{d'}((1-\beta)\tau)^{d'} \\
&= \Big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\Big)^{1/3}, \qquad \text{(A.35)}
\end{aligned}$$

which is strictly greater than $\left(\frac{9K^2(0)e}{144K^2(\tau)}\right)^{1/3}$.

At this point, it suffices for us to show that the condition $\varepsilon_{\text{CDF}} < 6|\mathcal{N}_Q(u)|$ holds. This inequality can be written as

$$|\mathcal{N}_Q(u)|^3 > \frac{9K^2(0)}{144K^2(\tau)} \log\Big(|\mathcal{N}_Q(u)|^3 \frac{144K^2(\tau)}{9K^2(0)}\Big). \tag{A.36}$$

Define

$$a := \frac{9K^2(0)}{144K^2(\tau)}, \qquad b := \frac{9K^2(0)}{144K^2(\tau)} \log\Big(\frac{144K^2(\tau)}{9K^2(0)}\Big) + \frac{18K^2(0)}{144K^2(\tau)}.$$

Then Lemma D.2(c) of Chen (2019) says that if $\frac{b}{a} + \log a > 1$ and

$$|\mathcal{N}_Q(u)|^3 \geq a\Big(1 + \sqrt{2\log(ae^{b/a-1})} + \log(ae^{b/a-1})\Big) = \Big(\frac{9K^2(0)}{144K^2(\tau)}(2+\sqrt{2})\Big)^{1/3}, \tag{A.37}$$

then

$$|\mathcal{N}_Q(u)|^3 \geq a\log(|\mathcal{N}_Q(u)|^3) + b = \frac{9K^2(0)}{144K^2(\tau)} \log\Big(|\mathcal{N}_Q(u)|^3 \frac{144K^2(\tau)}{9K^2(0)}\Big) + \frac{18K^2(0)}{144K^2(\tau)},$$

which implies that condition (A.36) holds. In fact, we have already shown that condition (A.37) holds (see inequality (A.35)). Thus, it suffices to show that $\frac{b}{a} + \log a > 1$. Indeed,

$$\frac{b}{a} + \log a = \frac{\frac{9K^2(0)}{144K^2(\tau)} \log\Big(\frac{144K^2(\tau)}{9K^2(0)}\Big) + \frac{18K^2(0)}{144K^2(\tau)}}{\frac{9K^2(0)}{144K^2(\tau)}} + \log\Big(\frac{9K^2(0)}{144K^2(\tau)}\Big) = 2 > 1.$$

At this point, we can conclude that condition (A.36) holds, which completes our justification that the sufficient conditions (A.34) hold. Thus, we can combine equation (A.31), inequality (A.32), and inequality (A.33) to get that

$$\text{term } (c) \leq \frac{1}{|\mathcal{N}_Q(u)|} \leq \frac{1}{\Xi_u} < \frac{2}{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}, \tag{A.38}$$

making use of event $\mathcal{E}_\beta(u)$ and Lemma 7.

Completing the proof Plugging the bounds on terms (a), (b), and (c) (given in inequalities (A.25), (A.30), and (A.38)) back into the union bound (A.24), we get

$$\begin{aligned}
&\mathbb{E}_{U_{1:n},Y_{1:n},D_{1:n}}[\mathbb{1}\{(\mathcal{E}_\beta(u) \cap \mathcal{E}_{\text{horizon}}(u) \cap \mathcal{E}_{\text{CDF}}(u))^c\}] \\
&\quad \leq \frac{8}{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))} + \frac{4}{\theta^2 n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))} + \frac{2}{n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))} \\
&\quad \leq \frac{14}{\theta^2 n\mathbb{P}_{\mathsf{U}}(B(u,(1-\beta)\tau))}.
\end{aligned}$$

■

Appendix B. Details on TUNA

Previously, Chen (2020) showed how to warm-start deep kernel survival analysis using any pre-trained kernel function, such as one obtained from decision trees or their ensembled variants (random forests, gradient tree boosting). However, Chen's strategy does not scale to large datasets (as we explain shortly). We first state Chen's strategy (in terms of pre-training data, although Chen originally stated this in terms of training data) and then we say how we incorporate it as a warm-start strategy to our own TUNA warm-start strategy described in Section 4.

Warm-start strategy by Chen (2020) Let $\mathbb{K}_0$ be a pre-trained kernel function (learned from, for instance, a random survival forest or XGBOOST). Then compute the $n_\circ$-by-$n_\circ$ matrix $\mathbf{K}$, where $\mathbf{K}_{i,j} = \mathbb{K}_0(X_i^\circ, X_j^\circ)$ for every pair of pre-training feature vectors X_i° and X_j°. Note that computing the matrix $\mathbf{K}$ is prohibitively expensive for large datasets, where even storing this whole matrix can be impractical. For the moment, suppose that we can compute and store this matrix. Next, if we are using a Gaussian kernel $\mathbb{K}(X_i^\circ, X_j^\circ) = \exp(-\|\phi(X_i^\circ) - \phi(X_j^\circ)\|_2^2)$, then by setting this expression equal to $\mathbf{K}_{i,j}$ and with a bit of algebra, we obtain

$$\|\phi(X_i^\circ) - \phi(X_j^\circ)\|_2 = \sqrt{\log \frac{1}{\mathbf{K}_{i,j}}} \quad \text{for all } i, j.$$

To prevent division by 0 in the log, we could for instance add a small constant to all values of $\mathbf{K}$ or clip values of $\mathbf{K}$ under a user-specified threshold to be equal to the threshold.

Next, we can use metric multidimensional scaling (MDS) (Borg and Groenen, 2005) to learn an embedding vector $\widetilde{X}_i^{\text{MDS}}$ for each pre-training feature X_i° such that $\|\widetilde{X}_i^{\text{MDS}} - \widetilde{X}_j^{\text{MDS}}\|_2 \approx \sqrt{\log \frac{1}{\mathbf{K}_{i,j}}}$ for all pairs i and j. Then for a base neural net ϕ with a user-specified architecture, we warm-start ϕ by minimizing the mean squared error loss

$$\frac{1}{n} \sum_{i=1}^{n} (\phi(X_i^\circ) - \widetilde{X}_i^{\text{MDS}})^2.$$

Effectively we are having the neural net ϕ approximate the MDS embedding, which approximates the Euclidean distances induced by the pre-trained kernel function under the assumption of a Gaussian kernel. This MDS-based strategy generalizes to other choices of kernel functions that can be written in terms of the distance function $\rho(x, x') = \|\phi(x) - \phi(x')\|_2$, so long as we can solve the equation $\mathbb{K}(X_i^\circ, X_j^\circ) = \mathbf{K}_{i,j}$ for $\rho(X_i^\circ, X_j^\circ)$; in fact, ρ could even be chosen to be non-Euclidean if generalized MDS is used, which handles non-Euclidean distances (Bronstein et al., 2006).

Incorporating Chen's warm-start strategy into TUNA To avoid having to compute the full $n_\circ$-by-$n_\circ$ kernel matrix $\mathbf{K}$ (and also having to solve MDS for such a large input matrix), we instead begin by only computing it over a subsample, much like Landmark Isomap (de Silva and Tenenbaum, 2002). In more detail, our full TUNA warm-start procedure is as follows:

1. Train a scalable tree ensemble (e.g., XGBOOST) on the pre-training data; denote its learned kernel function by $\mathbb{K}_0$. For a subset $\mathcal{S} \subseteq \{X_1^\circ, \ldots, X_{n_\circ}^\circ\}$ of the pre-training

feature vectors, let $\mathbf{K}_{\mathcal{S}}$ denote the $|\mathcal{S}|$-by-$|\mathcal{S}|$ Gram matrix formed so that the (i,j)-th entry is given by $\mathbb{K}_0(x_i^\circ, x_j^\circ)$ where x_i° and x_j° are the i-th and j-th pre-training feature vectors in $\mathcal{S}$ (with elements of $\mathcal{S}$ ordered arbitrarily).

1′. (MDS initialization by Chen (2020)) Take a subsample of size $n_{\text{subsample}}$ from the pre-training feature vectors (e.g., uniformly at random or using an ε-net). Denote the subsample by $\mathcal{S}_{\text{init}}$. Compute the gram matrix $\mathbf{K}_{\mathcal{S}_{\text{init}}}$. Use Chen's warm-start strategy restricted to pre-training data in $\mathcal{S}_{\text{init}}$ and using pre-trained kernel matrix $\mathbf{K}_{\mathcal{S}_{\text{init}}}$ to obtain an initial estimate of the base neural net ϕ.

2. For each minibatch consisting of pre-training feature vectors $x_1^\circ, \ldots, x_b^\circ$ where b is the batch size:

(a) Compute the batch's tree ensemble Gram matrix $\mathbf{K}_{\{x_1^\circ,\ldots,x_b^\circ\}}$ (defined in step 1) using $\mathbb{K}_0$.

(b) Compute the current neural net's Gram matrix estimate $\widehat{\mathbf{K}}_{\{x_1^\circ,\ldots,x_b^\circ\}}$, which has (i,j)-th entry given by $\mathbb{K}(x_i^\circ, x_j^\circ) = K(\|\phi(x_i^\circ) - \phi(x_j^\circ)\|_2^2)$.

(c) Let this minibatch's loss be the MSE loss (4.1) restricted to feature vectors of the current minibatch, i.e., the MSE loss between $\widehat{\mathbf{K}}_{\{x_1^\circ,\ldots,x_b^\circ\}}$ and $\mathbf{K}_{\{x_1^\circ,\ldots,x_b^\circ\}}$. Update parameters of neural net ϕ based on the gradient of this minibatch's loss.

In our experiments later, for simplicity, we construct $\mathcal{S}_{\text{init}}$ in step 1′ by taking a uniform subsample of the training data and set $n_{\text{subsample}} = 2048$.

By running minibatch gradient descent (step 2) for a number of epochs (we use 100 for SUPPORT and UNOS, and 10 for KKBOX), we obtain the initial neural net $\widehat{\phi}$ that we begin survival kernet training with.

Appendix C. Details on Experiments

We now provide additional details on data preprocessing (Appendix C.1), and on hyperparameter grids and optimization (Appendix C.2). An additional cluster visualization is in Appendix C.4.

C.1 Preprocessing Notes

Continuous features are standardized (subtract mean and divide by standard deviation). Categorical features are one-hot encoded unless they correspond to features with a clear ordering in which case they are converted to be on a scale from 0 to 1 (evenly spaced for where the levels are). The UNOS data preprocessing is a bit more involved and follows the steps of Yoon et al. (2018) and Lee et al. (2018). The KKBOX dataset is preprocessed via the `pycox` package (Kvamme et al., 2019).

C.2 Hyperparameter Grids (Including Neural Net Architecture Settings) and Optimization Details

We provide hyperparameter grids and optimization details in this section. Before doing so, we make a few remarks. First, we implement the elastic-net-regularized Cox model as well as all neural net models in PyTorch (Paszke et al., 2019) so that we can train these models via minibatch gradient descent, which is helpful due to the size of some of our datasets.

Note that XGBOOST supports survival analysis with a few losses. We specifically use the Cox loss. We train all models that are implemented in PyTorch (including ELASTIC-NET COX) using Adam (Kingma and Ba, 2015) with a budget of epochs 100 (except for the KKBOX dataset, where we use 10). Early stopping is used based on the validation set (no improvement in the best validation loss within 10 epochs). For simplicity, we always use a batch size of 1024 (from some preliminary experiments, we find that smaller batch sizes such as 128, 256, and 512 yield similar results for baselines but for survival kernets, a larger batch size appears to be helpful).

For the KKBOX dataset, because of how large it is, even computing the validation loss is computationally expensive (C^{td} index is a ranking metric that considers every pair of data points). To make training more practical, we use a subsampling strategy for computing the validation loss where we randomly partition the validation data into groups of size 2^{14}, compute the C^{td} index per group, and then average the indices computed across the groups. We empirically found this approximation to be close to the exact calculation and takes much less time to compute. For the test set as well as bootstrap confidence intervals, we do an exact calculation rather than using this subsampling strategy.

We now list hyperparameter grids of the different methods. Even more details are available in our code.

ELASTIC-NET COX:

- Learning rate: 1.0, 0.1, 0.01, 0.001, 0.0001
- Regularization weight: 0 (corresponds to the standard Cox model (Cox, 1972)), 0.01, 1
- Elastic-net knob for how much to use ℓ_1 regularization vs squared ℓ_2 regularization: 0 (ridge regression), 0.5 (evenly balance ℓ_1 and squared ℓ_2), 1 (lasso)

XGBOOST:

- Number of random features selected per node: $\frac{1}{2} \times$ sqrt of number of features, sqrt of number of features, $2 \times$ sqrt of number of features, use all features
- Learning rate "eta": 0.1, 0.3, 1
- Number of parallel trees: 1, 10
- Training data subsampling when growing trees: 0.6, 0.8
- Max depth: 3, 6, 9, 12
- Max number of rounds: 100 (for KKBOX, only 20)

DEEPSURV:

- Learning rate: 0.01, 0.001
- Number of hidden layers: 2, 4, 6
- Number of nodes per hidden layer: 32, 64, 128
- Nonlinear activation of hidden layers: ReLU
- Use batch norm after each hidden layer: yes

The final layer (a linear layer) is not a hidden layer and corresponds to a Cox model; it has 1 output node and no bias.

DEEPHIT has the same search grid as DEEPSURV and also additionally has the following:

- "alpha": 0, 0.001, 0.01
- "sigma": 0.1, 1
- Number of time steps to discretize to: 0 (i.e., use all unique observed times of death), 64, 128

Importantly, for discretizing time, we use evenly spaced quantiles of observed times (for details, see Section 3.1 of Kvamme and Borgan (2021)), which in some circumstances could result in the number of time steps used be *fewer* than what the user specifies. For example, when discretizing to 64 time steps, we take 64 evenly spaced quantiles of observed times. Suppose that 99.9% of the observed times happen to all be exactly 1 day whereas the other 0.1% of the observed times are exactly 2 days. Then the 64 evenly spaced quantiles would actually all correspond to 1 day as the observed time. Of course, if all the observed times are unique, then we would not have this sort of issue.

DCM has the same search grid as DEEPSURV and also additionally has the following:

- Number of Cox distributions: 3, 4, 5, 6
- Smoothing factor: 10^{-3}

The MLP base neural net's final number of output nodes is precisely the number of dimensions of the embedding space. We set this number to be the minimum of: (1) the number of nodes per hidden layer, and (2) the number of features in feature space (after preprocessing).

DKSA (we used the reference code by the original author specifically with random survival forest initialization; the only changes made were to get it to work with our experimental harness that loads in data, tunes hyperparameters, and gets test set metrics):

- Number of random features selected per node (for random survival forest warm-start): $\frac{1}{2} \times$ sqrt of number of features, sqrt of number of features, $2 \times$ sqrt of number of features
- Minimum data points per leaf (for random survival forest warm-start): 8, 32, 128
- Learning rate: 0.01, 0.001
- Number of hidden layers: 2, 4, 6
- Number of nodes per hidden layer: 32, 64, 128
- Nonlinear activation of hidden layers: ReLU
- Use batch norm after each hidden layer: yes
- Number of time steps to discretize to: 0 (i.e., use all unique observed times of death), 64, 128; we use the same discretization strategy stated above for DEEPHIT

The MLP base neural net's final number of output nodes is set the same way as described above for DCM.

Note that we always project the base neural net's output onto a hypersphere of radius $\sqrt{0.1}$ (this is mathematically equivalent to projecting to the unit hypersphere but changing the kernel function to instead be $K(u) = \exp(-u^2/0.1)$). In fact, this idea is commonly used in contrastive learning and the radius is a hyperparameter that can be tuned and is equal to the square root of two multiplied by what is commonly called the "temperature" hyperparameter (see for instance the papers by Wang and Isola (2020) and Liu et al. (2021)).

KERNET has the same search grid as DEEPSURV and also additionally has the following:

- η: 0, 0.001, 0.01 (defined in L_{rank} as part of equation (2.7); this hyperparameter is analogous to DEEPHIT's "alpha" hyperparameter)
- σ_{rank}: 0.1, 1 (also defined in L_{rank} as part of equation (2.7); this hyperparameter is analogous to DEEPHIT's "sigma" hyperparameter)
- Number of time steps to discretize to: 0 (i.e., use all unique observed times of death), 64, 128; we use the same discretization strategy stated above for DEEPHIT
- τ: $\sqrt{2\log(10)} \approx 2.146$ (we additionally set the maximum number of approximate nearest neighbors to be 128)
- β: 1/4, 1/2 (we construct ε-nets with $\varepsilon = \beta\tau$)

We set the MLP base neural net's final number of output nodes the same way as the DKSA baseline and again project the base neural net's output to a hypersphere with radius $\sqrt{0.1}$. From preliminary experiments, using a unit hypersphere (i.e., setting the radius to be equal to 1) typically resulted in worse validation accuracy scores.

TUNA-KERNET is the same as KERNET except that our TUNA warm-start strategy is used.

C.3 More Detailed Computation Time Breakdown Example

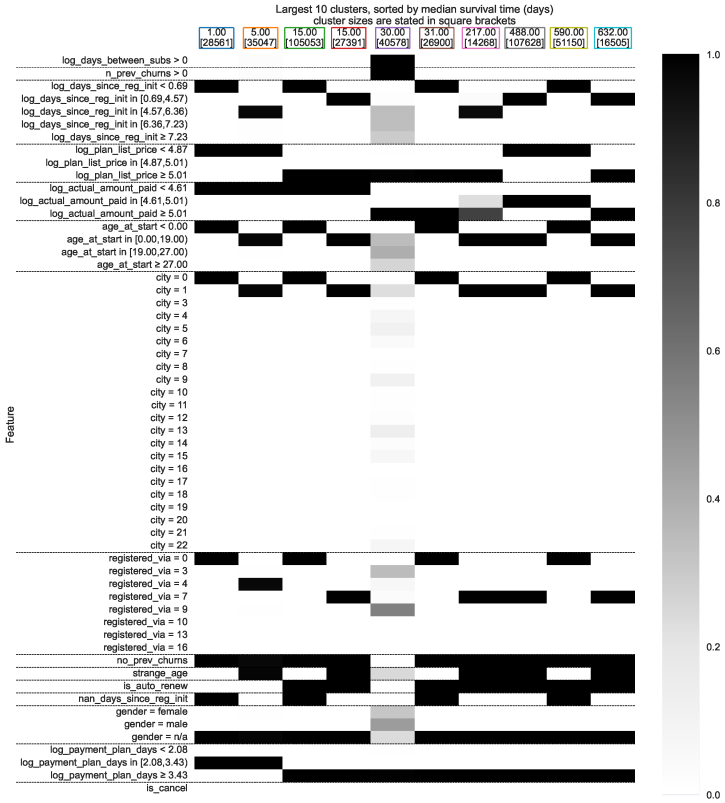
As an illustrative example, we give a more detailed breakdown of Table 5.4 where specifically for the step that fine-tunes the base neural net with the DKSA loss, we not only separate computation times with respect to β but we also separate the computation times with respect to the number of time steps that we discretize to. The resulting breakdown is shown in Table C.1. This same idea could of course be used to analyze the amount of time of other hyperparameters too. Specifically for TUNA-KERNET (NO SPLIT, SFT), we see that for the KKBOX dataset, discretizing using 64 or 128 time steps takes a similar amount of time (although using 128 time steps is of course a little more costly), while discretizing using all unique observed times of death is significantly more costly in time (but for KKBOX specifically, it turns out that discretizing using all unique observed times of death significantly improves accuracy). As a minor technical reminder, as stated in the Appendix C.2, when 64 time steps are used, we are not guaranteed to have exactly 64 time steps and we might have fewer (similarly for when 128 time steps are used).

C.4 Additional Cluster Visualization

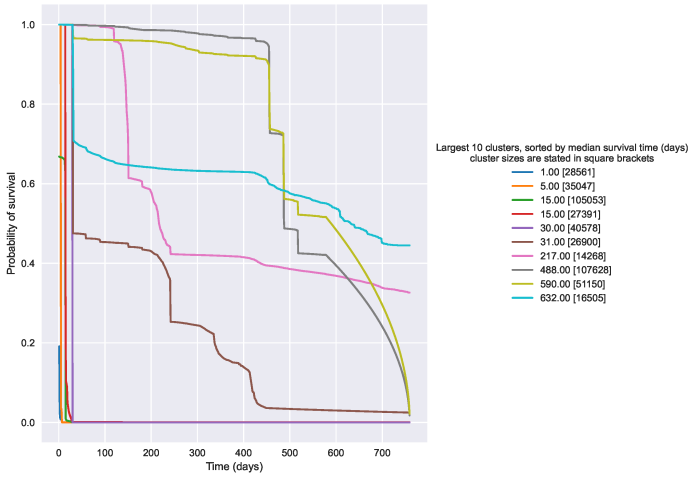
For the KKBOX dataset, we visualize the largest 10 clusters found by the final TUNA-KERNET (NO SPLIT, SFT) model in Figure 11. These largest 10 clusters only contain 28.7% of the proper training data.

C.5 Effect of Changing Threshold Distance τ

While the threshold distance τ for survival kernets could be tuned as part of hyperparameter tuning, we leave it fixed as we found in preliminary experiments that at least for the different values of τ that we tried, the qualitative flavor of the results did not drastically change. Specifically, we tried $\tau = \sqrt{\log 10}$ (i.e., training points only contribute to the prediction of a test point if they have kernel weight/similarity score at least 0.1 with the test point),



(a)



(b)

Figure 11: Visualization of the largest 10 clusters for the final TUNA-KERNET (NO SPLIT, SFT) model trained on the KKBOX dataset. Panel (a) shows a feature heatmap visualization. Panel (b) shows survival curves for the same clusters as in panel (a); the x-axis measures the number of days since an individual subscribed to the music streaming service.

Table C.1: Time breakdown in training the TUNA-KERNET (NO SPLIT, SFT) model on the KKBOX dataset for our experimental setup (mean ± standard deviation across 5 experimental repeats). The time breakdown for training TUNA-KERNET (NO SPLIT) is the same without the final summary fine-tuning time. Note that our TUNA warm-start strategy corresponds to the combination of the first two tasks listed below. Also, note that every task listed below involves tuning different hyperparameters. For details on hyperparameters, see Appendix C.2.

Task	Hyperparameter settings	Time (minutes)
Train XGBOOST model	192*	292.358 ± 1.943
Approximate XGBOOST kernel with base neural net	18†	111.987 ± 0.307
Fine-tune base neural net with DKSA loss ($\beta = 1/2$, discretize to 64 time steps)	12‡	163.928 ± 13.769
Fine-tune base neural net with DKSA loss ($\beta = 1/2$, discretize to 128 time steps)	12‡	170.394 ± 16.056
Fine-tune base neural net with DKSA loss ($\beta = 1/2$, discretize to all unique observed times of death)	12‡	459.561 ± 24.087
Fine-tune base neural net with DKSA loss ($\beta = 1/4$, discretize to 64 time steps)	12‡	189.322 ± 10.688
Fine-tune base neural net with DKSA loss ($\beta = 1/4$, discretize to 128 time steps)	12‡	196.903 ± 10.287
Fine-tune base neural net with DKSA loss ($\beta = 1/4$, discretize to all unique observed times of death)	12‡	504.150 ± 15.288
Summary fine-tuning	2§	471.814 ± 37.123
Total		2560.418 ± 65.785

* corresponds to the full XGBOOST hyperparameter grid that we use
† base neural net hyperparameters (number of hidden layers, nodes per hidden layer), learning rate
‡ survival loss hyperparameters (η and σ_{rank} from equation (2.7)), learning rate
§ learning rate

Table C.2: *Validation* set C^{td} indices (mean ± standard deviation across 5 experimental repeats) for TUNA-KERNET (NO SPLIT, SFT) using different threshold distances.

Threshold distance	Dataset			
	ROTTERDAM/GBSG	SUPPORT	UNOS	KKBOX
$\tau = \sqrt{\log 10}$	0.6852 ± 0.0212	0.6463 ± 0.0074	0.6134 ± 0.0035	0.9047 ± 0.0008
$\tau = \sqrt{2 \log 10}$	0.6874 ± 0.0198	0.6483 ± 0.0067	**0.6156** ± 0.0022	**0.9052** ± 0.0005
$\tau = \sqrt{4 \log 10}$	**0.6887** ± 0.0239	**0.6484** ± 0.0055	0.6135 ± 0.0026	0.9030 ± 0.0043

$\tau = \sqrt{2 \log 10}$ (minimum kernel weight 0.01), and $\tau = \sqrt{4 \log 10}$ (minimum kernel weight 0.0001). The validation set (and not test set) C^{td} indices are shown in Table C.2.

For example, looking specifically at the TUNA-KERNET (NO SPLIT, SFT) model, findings (a) and (b) in Section 5.1 still hold when we instead use $\tau = \sqrt{\log 10}$ (i.e., training points only contribute to the prediction of a test point if they have kernel weight/similarity score at least 0.1 with the test point) or $\tau = \sqrt{4 \log 10}$ (minimum kernel weight 0.0001); look at Table C.3 and compare the achieved test set C^{td} indices with those of the baseline models in Table 5.2.

Table C.3: Test set C^{td} indices (mean ± standard deviation across 5 experimental repeats) for TUNA-KERNET (NO SPLIT, SFT) using different threshold distances.

Threshold distance	Dataset			
	ROTTERDAM/GBSG	SUPPORT	UNOS	KKBOX
$\tau = \sqrt{\log 10}$	0.6708 ± 0.0098	0.6373 ± 0.0047	0.6175 ± 0.0059	0.9051 ± 0.0006
$\tau = \sqrt{2 \log 10}$	0.6719 ± 0.0135	0.6426 ± 0.0045	0.6211 ± 0.0025	0.9057 ± 0.0003
$\tau = \sqrt{4 \log 10}$	0.6706 ± 0.0124	0.6403 ± 0.0031	0.6181 ± 0.0026	0.9035 ± 0.0042

References

Anish Agarwal, Devavrat Shah, Dennis Shen, and Dogyoon Song. On robustness of principal component regression. *Journal of the American Statistical Association*, 116(536):1731–1745, 2021.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *ACM SigKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Symposium on Theory of Computing*, 2015.

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Advances in Neural Information Processing Systems*, 2015.

Laura Antolini, Patrizia Boracchi, and Elia Biganzoli. A time-dependent discrimination index for survival data. *Statistics in Medicine*, 24:3927–3944, 2005.

Timothy R. Asmis, Keyue Ding, Lesley Seymour, Frances A. Shepherd, Natasha B. Leighl, Tim L. Winton, Marlo Whitehead, Johanna N. Spaans, Barbara C. Graham, and Glenwood D. Goss. Age and comorbidity as independent prognostic factors in the treatment of non–small-cell lung cancer: A review of national cancer institute of canada clinical trials group trials. *Journal of Clinical Oncology*, 26(1):54–59, 2008.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(46):1345–1382, 2005.

Shai Ben-David, Dávid Pál, and Hans Ulrich Simon. Stability of k-means clustering. In *International Conference on Computational Learning Theory*, 2007.

Rudolf Beran. Nonparametric regression with randomly censored survival data. *Technical report, University of California, Berkeley*, 1981.

Gérard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(38):1063–1095, 2012.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Science & Business Media, 2005.

Leo Breiman. Some infinity theory for predictor ensembles. *Technical report 577, Statistics Department, University of California, Berkeley*, 2000.

Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006.

Charles C. Brown. On the use of indicator variables for studying the time-dependence of parameters in a response-time model. *Biometrics*, 31(4):863–872, 1975.

Gaëlle Chagny and Angelina Roche. Adaptive and minimax estimation of the cumulative distribution function given a functional covariate. *Electronic Journal of Statistics*, 8(2): 2352–2404, 2014.

Paidamoyo Chapfuwa, Chenyang Tao, Chunyuan Li, Courtney Page, Benjamin Goldstein, Lawrence Carin Duke, and Ricardo Henao. Adversarial time-to-event modeling. In *International Conference on Machine Learning*, 2018.

Paidamoyo Chapfuwa, Chunyuan Li, Nikhil Mehta, Lawrence Carin, and Ricardo Henao. Survival cluster analysis. In *Conference on Health, Inference, and Learning*, 2020.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, 2019.

George H. Chen. Nearest neighbor and kernel survival analysis: Nonasymptotic error bounds and strong consistency rates. In *International Conference on Machine Learning*, 2019.

George H. Chen. Deep kernel survival analysis and subject-specific survival time prediction intervals. In *Machine Learning for Healthcare Conference*, 2020.

George H. Chen and Devavrat Shah. Explaining the success of nearest neighbor methods in prediction. *Foundations and Trends® in Machine Learning*, 10(5-6):337–588, 2018.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *ACM SigKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

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

Dorota M. Dabrowska. Uniform consistency of the kernel conditional Kaplan-Meier estimate. *The Annals of Statistics*, 17(3):1157–1167, 1989.

Dominic Danks and Christopher Yau. Derivative-based neural modelling of cumulative distribution functions for survival analysis. In *International Conference on Artificial Intelligence and Statistics*, 2022.

Vin de Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, 2002.

Misha Denil, David Matheson, and Nando de Freitas. Narrowing the gap: Random forests in theory and in practice. In *International Conference on Machine Learning*, 2014.

Anne I. Dipchand. Current state of pediatric cardiac transplantation. *Annals of Cardiothoracic Surgery*, 7(1):31–55, 2018.

Matthew Engelhard, Samuel Berchuck, Joshua D'Arcy, and Ricardo Henao. Neural conditional event time models. In *Machine Learning for Healthcare Conference*, 2020.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, 1996.

John A. Foekens, Harry A. Peters, Maxime P. Look, Henk Portengen, Manfred Schmitt, Michael D. Kramer, Nils Brünner, Fritz Jänicke, Marion E. Meijer-van Gelder, Sonja C. Henzen-Logmans, Wim L. J. van Putten, and Jan G. M. Klijn. The urokinase system of plasminogen activation and prognosis in 2780 breast cancer patients. *Cancer Research*, 60(3):636–643, 2000.

Stephane Fotso. Deep neural networks for survival analysis based on a multi-task framework. *arXiv preprint arXiv:1801.05512*, 2018.

Pasi Fränti and Sami Sieranoja. How much can k-means be improved by using better initialization and repeats? *Pattern Recognition*, 93:95–112, 2019.

Charles Frey, Hong Zhou, Danielle Harvey, and Richard H. White. Co-morbidity is a strong predictor of early death and multi-organ system failure among patients with acute pancreatitis. *Journal of Gastrointestinal Surgery*, 11(6):733–742, 2007.

Michael F. Gensheimer and Balasubramanian Narasimhan. A scalable discrete-time survival model for neural networks. *PeerJ*, 7:e6257, 2019.

Eleonora Giunchiglia, Anton Nemchenko, and Mihaela van der Schaar. RNN-SURV: A deep recurrent model for survival analysis. In *International Conference on Artificial Neural Networks*, 2018.

Mark Goldstein, Xintian Han, Aahlad Puli, Adler Perotte, and Rajesh Ranganath. X-CAL: Explicit calibration for survival analysis. In *Advances in Neural Information Processing Systems*, 2020.

Humza Haider, Bret Hoehn, Sarah Davis, and Russell Greiner. Effective ways to build and evaluate individual survival distributions. *Journal of Machine Learning Research*, 21(85): 1–63, 2020.

Steve Hanneke, Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Universal Bayes consistency in metric spaces. *The Annals of Statistics*, 49(4):2129–2150, 2021.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.)*. Springer, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *IEEE International Conference on Computer Vision*, 2015.

John D. Kalbfleisch and Ross L. Prentice. *The Statistical Analysis of Failure Time Data*. Wiley, 1980.

Edward L. Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958.

Jared L. Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(24), 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

William A. Knaus, Frank E. Harrell, Joanne Lynn, Lee Goldman, Russell S. Phillips, Alfred F. Connors, Neal V. Dawson, William J. Fulkerson, Robert M. Califf, Norman Desbiens, Peter Layde, Robert K. Oye, Paul E. Bellamy, Rosemarie B. Hakim, and Douglas P. Wagner. The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122(3):191–203, 1995.

Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Nearest-neighbor sample compression: Efficiency, consistency, infinite dimensions. In *Advances in Neural Information Processing Systems*, 2017.

Samory Kpotufe and Nakul Verma. Time-accuracy tradeoffs in kernel prediction: controlling prediction quality. *Journal of Machine Learning Research*, 18(44):1–29, 2017.

Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *IEEE Symposium on Foundations of Computer Science*, 2010.

Håvard Kvamme and Ørnulf Borgan. Continuous and discrete-time survival prediction with neural networks. *Lifetime Data Analysis*, 27:710–736, 2021.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and Cox regression. *Journal of Machine Learning Research*, 20(129):1–30, 2019.

Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

Changhee Lee, William R. Zame, Jinsung Yoon, and Mihaela van der Schaar. DeepHit: A deep learning approach to survival analysis with competing risks. In *AAAI Conference on Artificial Intelligence*, 2018.

Linhong Li, Ren Zuo, Amanda Coston, Jeremy C. Weiss, and George H. Chen. Neural topic models with survival supervision: Jointly predicting time-to-event outcomes and learning how clinical features relate. In *International Conference on Artificial Intelligence in Medicine*, 2020.

Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. Learning with hyperspherical uniformity. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Yu A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

Laura Manduchi, Ričards Marcinkevičs, Michela C. Massi, Thomas Weikert, Alexander Sauter, Verena Gotta, Timothy Müller, Flavio Vasella, Marian C. Neidert, Marc Pfister, Bram Stieltjes, and Julia E. Vogt. A deep variational approach to clustering survival data. In *International Conference on Learning Representations*, 2022.

Chirag Nagpal, Steve Yadlowsky, Negar Rostamzadeh, and Katherine Heller. Deep Cox mixtures for survival regression. In *Machine Learning for Healthcare Conference*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Liudmila Prokhorenkova and Aleksandr Shekhovtsov. Graph-based nearest neighbor search: From practice to theory. In *International Conference on Machine Learning*, 2020.

Alexander Rakhlin and Andrea Caponnetto. Stability of k-means clustering. In *Advances in Neural Information Processing Systems*, 2006.

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. In *Machine Learning for Healthcare Conference*, 2016.

Mark J. Russo, Kimberly N. Hong, Ryan R. Davies, Jonathan M. Chen, Donna M. Mancini, Mehmet C. Oz, Eric A. Rose, Annetine Gelijns, and Yoshifumi Naka. The effect of body mass index on survival following heart transplantation: do outcomes support consensus guidelines? *Annals of Surgery*, 251(1):144–152, 2010.

M. Schumacher, G. Bastert, H. Bojar, K. Huebner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. L. Neumann, and H. F. Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. german breast cancer study group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

Victoria Sopik and Steven A. Narod. The relationship between tumour size, nodal status and distant metastases: on the origins of breast cancer. *Breast Cancer Research and Treatment*, 170(3):647–656, 2018.

Weijing Tang, Jiaqi Ma, Qiaozhu Mei, and Ji Zhu. SODEN: A scalable continuous-time survival model through ordinary differential equation networks. *Journal of Machine Learning Research*, 23(34):1–29, 2022.

Elmer E. van Eeghen, Sandra D. Bakker, Aart van Bochove, and Ruud J. L. F. Loffeld. Impact of age and comorbidity on survival in colorectal cancer. *Journal of Gastrointestinal Oncology*, 6(6):605–612, 2015.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Ulrike von Luxburg. Clustering stability: An overview. *Foundations and Trends® in Machine Learning*, 2(3):235–274, 2010.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 2020.

Zhiliang Wu, Yinchong Yang, Peter A. Fasching, and Volker Tresp. Uncertainty-aware time-to-event prediction using deep kernel accelerated failure time models. In *Machine Learning for Healthcare Conference*, 2021.

Jinsung Yoon, William R. Zame, Amitava Banerjee, Martin Cadeiras, Ahmed M. Alaa, and Mihaela van der Schaar. Personalized survival predictions via trees of predictors: An application to cardiac transplantation. *PloS one*, 13(3):e0194985, 2018.

He Zhao, Dinh Phung, Viet Huynh, Yuan Jin, Lan Du, and Wray Buntine. Topic modelling meets deep neural networks: A survey. In *International Joint Conference on Artificial Intelligence*, 2021.

Qixian Zhong, Jonas Mueller, and Jane-Ling Wang. Deep extended hazard models for survival analysis. *Advances in Neural Information Processing Systems*, 2021.

Qixian Zhong, Jonas Mueller, and Jane-Ling Wang. Deep learning for the partially linear Cox model. *The Annals of Statistics*, 50(3):1348–1375, 2022.