Deep Kernel Aalen-Johansen Estimator: An Interpretable and Flexible Neural Net Framework for Competing Risks

Xiaobin Shen* XIAOBINS@ANDREW.CMU.EDU
George H. Chen* GEORGECHEN@CMU.EDU
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We propose an interpretable deep competing risks model called the Deep Kernel Aalen-Johansen (DKAJ) estimator, which generalizes the classical Aalen-Johansen nonparametric estimate of cumulative incidence functions (CIFs). Each data point (e.g., patient) is represented as a weighted combination of clusters. If a data point has nonzero weight only for one cluster, then its predicted CIFs correspond to those of the classical Aalen-Johansen estimator restricted to data points from that cluster. These weights come from an automatically learned kernel function that measures how similar any two data points are. On four standard competing risks datasets, we show that DKAJ is competitive with state-of-the-art baselines while being able to provide visualizations to assist model interpretation.

Keywords: survival analysis, competing risks, neural networks, interpretability

Data and Code Availability We use standard publicly available competing risks benchmark datasets: PBC (Fleming and Harrington, 1991), Framingham (Kannel and McGee, 1979), SEER[1], and the synthetic dataset by Lee et al. (2018). Our code is available at: https://github.com/xiaobin-xs/dkaj

Institutional Review Board (IRB) This work does not require IRB approval as we analyze existing publicly available datasets.

1. Introduction

Various health applications involve reasoning about the amount of time that will elapse before a critical event happens, where there could be different types of critical events and we would like to know which type happens earliest and when. For example, for a patient in a hospital, we may want to model their time until either discharge or in-hospital mortality, where only one of these happens earliest. This is a well-studied problem referred to as the *competing risks* setup in survival analysis literature (e.g., see Chapter 8 of the textbook by Kalbfleisch and Prentice (2002)).

Despite many recent methodological advances in developing flexible competing risks models, the bulk of the methods developed are not easy to interpret. For example, deep competing risks models like DeepHit (Lee et al., 2018), DeSurv (Danks and Yau, 2022), and Neural Fine-Gray (Jeanselme et al., 2023) are not designed to be straightforward to interpret. Similarly, gradient boosted trees for competing risks (Alberge et al., 2025) are also not easy to interpret as soon as there are many trees each with many leaves. Instead, for competing risks, if one wants to use a model that is inherently interpretable, the standard approach is still to use the classical Fine and Gray model (Fine and Gray, 1999), the cause-specific Cox model (Prentice et al., 1978), or slight variants of either of these.[2]

In this paper, we propose a new deep competing risks model that aims to be interpretable. We build upon the classical Aalen-Johansen (AJ) estimator (Aalen and Johansen, 1978), which is a nonparametric approach developed for the competing risks setup that estimates population-level quantities. We show how to adapt the key ideas of the AJ estimator to instead produce predicted risks of different competing events at the individual data point level. What we get is a "conditional" AJ estimator, where we are

* These authors contributed equally.
1. https://seer.cancer.gov/
2. Despite its name, the Neural Fine-Gray model (Jeanselme et al., 2023) does not closely resemble the original Fine and Gray model (Fine and Gray, 1999) and is instead a competing risks extension of SuMo-net (Rindt et al., 2022); i.e., Neural Fine-Gray does not inherit the interpretation advantages of the original Fine and Gray model.

conditioning on a test point's feature vector. A non-deep-learning-based conditional AJ estimator has recently been studied theoretically (Bladt and Furrer, 2025). Our paper is the first that we are aware of that develops a deep-learning-based kernel AJ estimator.

Our deep kernel Aalen-Johansen (DKAJ) estimator represents each data point as a weighted combination of clusters (each cluster corresponds to a subset of training patients). Per cluster, we store summary information that corresponds to an AJ estimator restricted to data points (i.e., patients) in the cluster. Consequently, existing research on interpreting the AJ estimator can be used. Note that this interpretation is not "post-hoc" in that when making a prediction for a test point, if the test point is represented as a weighted combination that puts nonzero weight only on a single cluster, its prediction does correspond to the AJ estimator for that cluster. The weights in these weighted combinations are based on an automatically learned kernel function that measures how similar any two data points are.

From a technical viewpoint, the AJ estimator is the competing risks analogue of the classical Kaplan-Meier (KM) estimator (Kaplan and Meier, 1958) that is meant for standard survival analysis without competing risks. Similar to how deep kernel KM estimators were developed by Chen (2020, 2024b), our proposed method could be thought of as extending Chen's earlier deep kernel KM estimator approach to the competing risks setting, replacing the KM estimator with the AJ estimator. In fact, when there is only a single competing event type, our proposed method becomes the existing *survival kernets* approach by Chen (2024b) that does not handle competing risks.

2. Background

We briefly review the statistical setup for survival analysis with competing risks (Section 2.1) and the Aalen-Johansen (AJ) estimator (Section 2.2), both of which we present at a level of detail sufficient for understanding our deep kernel extension of the AJ estimator. Although our work generalizes the survival kernets method by Chen (2024b), we do not review it in this section and instead explain how it is a special case of our proposed method in Section 3.

Notation We use uppercase letters (e.g., X) to denote random variables and lowercase letters (e.g. x) to denote realizations of random variables as well as constants, although we also use uppercase S to denote a survival function, as is standard in survival analysis literature. For any positive integer ℓ, we use the notation $[\ell] \triangleq \{1, 2, \ldots, \ell\}$. Meanwhile, we use $\mathbb{1}\{\cdot\}$ to denote the indicator function that is 1 when its argument is true and 0 otherwise.

2.1. Statistical Setup

We use the setup in Section 6.1 of Chen (2024a). We assume that there are m critical event types that are mutually exhaustive, and that we have access to i.i.d. training data $(X_1, Y_1, \Delta_1), \ldots, (X_n, Y_n, \Delta_n)$, where for the i-th subject, $X_i \in \mathcal{X}$ denotes the subject's feature vector[3], $Y_i \in [0, \infty)$ denotes the time until the earliest critical event that happens or censoring, and $\Delta_i \in \{0, 1, \ldots, m\}$ is an event indicator stating which critical event type happens earliest (the special value of 0 means that censoring happens earliest). Each point (X_i, Y_i, Δ_i) is generated as follows:

1. Sample feature vector X_i from $\mathbb{P}_{\mathsf{X}}$.
2. Sample the time $T_{i,\delta}$ until each event $\delta \in [m]$ happens. We do this jointly, i.e., we sample length-m vector $\boldsymbol{T}_i \triangleq (T_{i,1}, T_{i,2}, \ldots, T_{i,m})$ from distribution $\mathbb{P}_{\mathsf{T}|\mathsf{X}}(\cdot|X_i)$, which we assume to be absolutely continuous (so ties among the m entries in $\boldsymbol{T}_i$ happen with probability 0). We denote the time of the earliest event as $T_i \triangleq \min_{\delta\in[m]} T_{i,\delta}$, and the earliest event that happens as $\Delta_i^* \triangleq \arg\min_{\delta\in[m]} T_{i,\delta}$.
3. Sample censoring time C_i from absolutely continuous distribution $\mathbb{P}_{\mathsf{C}|\mathsf{X}}(\cdot|X_i)$.
4. Set $Y_i \triangleq \min\{T_i, C_i\}$, and
$$\Delta_i \triangleq \begin{cases} 0 & \text{if } Y_i = C_i, \\ \Delta_i^* & \text{otherwise.} \end{cases}$$

Steps 2 and 3 imply that conditioned on X_i, random variables $\boldsymbol{T}_i$ and C_i are independent.

Prediction task We assume that any test point is sampled using only the first two steps of the generative process for training data (i.e., we do not model a test point being censored). Namely:

1. Sample test feature vector X from $\mathbb{P}_{\mathsf{X}}$.
2. Sample $\boldsymbol{T} \triangleq (T_1, \ldots, T_m)$ from $\mathbb{P}_{\mathsf{T}|\mathsf{X}}(\cdot|X)$, and set $T \triangleq \min_{\delta\in[m]} T_\delta$, and $\Delta^* \triangleq \arg\min_{\delta\in[m]} T_\delta$.

We aim to predict the *cumulative incidence function* (CIF), which is specific to each event type $\delta \in [m]$:

$$F_\delta(t|x) \triangleq \mathbb{P}(T \le t, \Delta^* = \delta \mid X = x) \quad \text{for } t \ge 0. \quad (1)$$

3. More generally, throughout the paper, the "feature vector" refers to time-independent covariates available at time 0 per data point. When non-tabular data are used (e.g., a medical image or ECG snippet), we assume that they are preprocessed into fixed-length embedding vectors that are treated as the "feature vectors". Dynamic prediction with time-varying covariates is beyond the scope of this paper and left for future work.

There are alternative ways to represent this CIF. For example, we can back out the CIF by knowing the *event-specific hazard function* (also called the *cause-specific hazard function* in the literature)

$$\lambda_\delta(t|x) \triangleq \lim_{h\downarrow 0} \frac{\mathbb{P}\big(t \le T < t+h, \Delta^* = \delta \,\big|\, T \ge t, X = x\big)}{h} = \frac{f_\delta(t|x)}{S(t|x)},$$

where $f_\delta(t|x) \triangleq \frac{\mathrm{d}}{\mathrm{d}t} F_\delta(t|x)$ is the event-specific subdensity function, and

$$S(t|x) \triangleq \mathbb{P}(T > t \,|\, X = x) = \exp\Bigg(-\sum_{\delta=1}^{m} \int_0^t \lambda_\delta(u|x)\mathrm{d}u\Bigg) \tag{2}$$

is called the *survival function*. In particular, if we know the event-specific hazard function $\lambda_\delta(t|x)$ for all events $\delta \in [m]$, then we can obtain $S(t|x)$ using equation (2), and subsequently recover $f_\delta(t|x) = \lambda_\delta(t|x)S(t|x)$. Finally, we can back out CIF $F_\delta(t|x)$ by integration: $F_\delta(t|x) = \int_0^t f_\delta(u|x)\mathrm{d}u$.

Likelihood function The standard competing risks likelihood function that does not depend on the censoring distribution is given by

$$\mathcal{L} = \prod_{i=1}^{n} \big(\lambda_{\Delta_i}(Y_i|X_i)\big)^{\mathbb{1}\{\Delta_i \neq 0\}} S(Y_i|X_i). \tag{3}$$

For details, see Kalbfleisch and Prentice (2002, Section 8.2.3). We rely on this likelihood function later as we use a maximum likelihood approach.

2.2. Aalen-Johansen (AJ) Estimator

We now describe a simplified version of the AJ estimator (as stated by Edwards et al. (2016) and implemented in the `lifelines` software package (Davidson-Pilon, 2019)). For a more general introduction to the AJ estimator (for multistate processes), see, for instance, Chapter 3 of Cook and Lawless (2018). The simplified AJ estimator we consider estimates the population-level CIF $F_\delta^{\mathrm{pop}}(t) \triangleq \mathbb{P}(T \le t, \Delta^* = \delta)$; this is "population-level" since it does not condition on X as in equation (1).

Let $0 < t_1 < t_2 < \cdots < t_L$ denote the unique times in which any critical event happens in the training data (censoring is not a critical event). Next, we denote the number of critical events of type $\delta \in [m]$ that occur at time t_ℓ (with $\ell \in [L]$) by

$$d_{\delta,\ell} \triangleq \sum_{j=1}^{n} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\}, \tag{4}$$

and the number of subjects "at risk" (who could possibly still experience any critical event) at time t_ℓ by

$$n_\ell \triangleq \sum_{j=1}^{n} \mathbb{1}\{Y_j \ge t_\ell\}. \tag{5}$$

Then the simplified AJ estimator of $F_\delta^{\mathrm{pop}}(t)$ is

$$\widehat{F}_\delta^{\mathrm{AJ}}(t) \triangleq \sum_{\ell=1}^{L} \frac{\mathbb{1}\{t_\ell \le t\} d_{\delta,\ell}}{n_\ell} \widehat{S}^{\mathrm{KM}}(t_{\ell-1}) \text{ for } t \ge 0, \tag{6}$$

where $t_0 \triangleq 0$, and $\widehat{S}^{\mathrm{KM}}$ is the classical Kaplan-Meier estimator (Kaplan and Meier, 1958):[4]

$$\widehat{S}^{\mathrm{KM}}(t) \triangleq \prod_{\ell=1}^{L} \Big(1 - \frac{\sum_{\delta=1}^{m} d_{\delta,\ell}}{n_\ell}\Big)^{\mathbb{1}\{t_\ell \le t\}} \text{ for } t \ge 0. \tag{7}$$

The key observation we exploit in our derivation of our DKAJ estimator is that the simplified AJ estimator (6) can be derived by maximizing the likelihood $\mathcal{L}$ in equation (3), with some pre- and post-processing.

Proposition 1 *Using the notation above where $0 < t_1 < \cdots < t_L$ denote the unique times in which any critical event happens, suppose that we parameterize the event-specific hazard function to be piecewise constant on the L intervals $(t_0, t_1], (t_1, t_2], \ldots, (t_{L-1}, t_L]$:*

$$\lambda_\delta(t|x) \triangleq \begin{cases} \dfrac{\phi_{\delta,\ell}}{t_\ell - t_{\ell-1}} & \text{if } t \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

where $\phi_{\delta,\ell} \in [0, \infty)$ for $\delta \in [m], \ell \in [L]$ are the parameters. These parameters do not depend on x, so the model here is at the population level. Thus, only for this section, we abbreviate $\lambda_\delta(t) \triangleq \lambda_\delta(t|x)$.

*Suppose that we preprocess our training data $(X_1, Y_1, \Delta_1), \ldots, (X_n, Y_n, \Delta_n)$ so that any point that is censored (so that $\Delta_i = 0$) has its observed time Y_i replaced by the latest time in which any critical event happened **prior to** Y_i (if no critical event happened before Y_i, then set $Y_i = 0$).[5] After doing this preprocessing, the choice of parameters $\phi_{\delta,\ell}$ that maximizes the likelihood of equation (3) is given by*

$$\widehat{\phi}_{\delta,\ell} = \frac{d_{\delta,\ell}}{n_\ell} \quad \text{for } \delta \in [m], \ell \in [L].$$

Thus, the maximum likelihood estimator for the event-specific hazard function is—by plugging $\widehat{\phi}_{\delta,\ell}$

4. The Kaplan-Meier estimator is for the standard survival analysis setup without competing risks (equivalent to the competing risks setup with a single event type); here, all m event types are lumped together into a single critical event type corresponding to any of the critical events happening.
5. This preprocessing procedure is used by Breslow to derive the Breslow estimator (Breslow, 1972) for the Cox proportional hazards model (Cox, 1972).

into $\phi_{\delta,\ell}$ in equation (8)—given by

$$\widehat{\lambda}_\delta(t) \triangleq \begin{cases} \dfrac{d_{\delta,\ell}}{(t_\ell - t_{\ell-1})n_\ell} & \text{if } t \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The estimated CIF in equation (6) can be obtained via a form of interpolation based on this piecewise-constant maximum likelihood estimator of $\lambda_\delta(t|x)$. We defer details of this postprocessing step (interpolation) and the proof of this proposition to Appendix A.

3. Deep Kernel Aalen-Johansen

We now derive our neural net extension of the AJ estimator that uses a kernel function $K : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ to measure how similar any two input points are (for two data points with feature vectors $x, x' \in \mathcal{X}$, their similarity score is given by $K(x, x')$, where higher means "more similar"). We refer to our resulting approach as the *deep kernel Aalen-Johansen* (DKAJ) estimator. This section is organized as follows:

- (Section 3.1) We first explain how we parameterize the event-specific hazard function in terms of kernel function K that is specified as a neural net, and how to maximize likelihood function (3) to train neural net parameters. This training procedure amounts to learning an embedding space to represent data points (e.g., patients) in.
- (Section 3.2) We then present our DKAJ model, which first runs the neural net training from Section 3.1. Afterwards, we run exemplar-based clustering in the learned embedding space, and compute summary information per cluster. Any data point is then represented as a weighted combination of clusters. We explain why the simplified AJ estimator (6) as well as survival kernets (Chen, 2024b) are special cases of DKAJ, and we also discuss model interpretation.

3.1. Neural Net Formulation and Loss

The key idea is that we modify the parameterization in equation (8) to depend on feature vectors. Reusing notation from Section 2.2 where $0 < t_1 < \cdots < t_L$ denote the unique times in which any critical event occurs in training data, we use the parameterization

$$\lambda_\delta(t|x) \triangleq \begin{cases} \dfrac{\psi_{\delta,\ell}(x)}{t_\ell - t_{\ell-1}} & \text{if } t \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where

$$\psi_{\delta,\ell}(x) \triangleq \frac{d_{\delta,\ell}(x)}{n_\ell(x)}, \tag{11}$$

$$d_{\delta,\ell}(x) \triangleq \sum_{j=1}^{n} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\} K(x, X_j), \tag{12}$$

$$n_\ell(x) \triangleq \sum_{j=1}^{n} \mathbb{1}\{Y_j \geq t_\ell\} K(x, X_j). \tag{13}$$

Notice that $d_{\delta,\ell}(x)$ (equation (12)) and $n_\ell(x)$ (equation (13)) generalize $d_{\delta,\ell}$ (equation (4)) and n_ℓ (equation (5)), respectively. In particular, we use the kernel function to weight each training point X_i by how similar it is to feature vector x. If the kernel function were to always output 1, then we would just obtain the maximum likelihood estimate from equation (9).

Next, we parameterize the kernel function K in the same manner as Chen (2020):

$$K(x, x') \triangleq \exp(-\|f(x;\theta) - f(x';\theta)\|^2), \tag{14}$$

where $\|\cdot\|$ is Euclidean distance, and $f(\cdot;\theta) : \mathcal{X} \to \mathbb{R}^d$ is a user-specified neural net such as a multilayer perceptron, with all neural net parameters collected into the variable θ (standard strategies can be used such as choosing f to be a CNN or vision transformer if "feature vectors" are instead images, etc).

At this point, combining equations (11), (12), (13), and (14), we get

$$\psi_{\delta,\ell}(x;\theta) = \frac{\sum_{j=1}^{n} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\} \exp(-\|f(x;\theta) - f(X_j;\theta)\|^2)}{\sum_{j=1}^{n} \mathbb{1}\{Y_j \geq t_\ell\} \exp(-\|f(x;\theta) - f(X_j;\theta)\|^2)}, \tag{15}$$

where we now write "$\psi_{\delta,\ell}(x;\theta)$" instead of only "$\psi_{\delta,\ell}(x)$" to emphasize the dependence on θ.

We state the log likelihood next with the help of equation (15). Let $\kappa(Y_i) \in \{0, 1, \ldots, L\}$ denote the time index corresponding to Y_i (where we first apply the same preprocessing to censored times as mentioned in Proposition 1). Then plugging in the event-specific hazard function $\lambda_\delta(t|x)$ into the log of the likelihood function $\mathcal{L}$ from equation (3), after a short derivation (for details, see Appendix B), we obtain

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{n} \sum_{\delta=1}^{m} \Bigg[\mathbb{1}\{\Delta_i = \delta\} \log \psi_{\delta,\kappa(Y_i)}(X_i;\theta) - \sum_{\ell=1}^{\kappa(Y_i)} \psi_{\delta,\ell}(X_i;\theta) \Bigg] + \text{constant}. \tag{16}$$

Maximizing this likelihood is problematic since the i-th training point's loss term uses a predicted hazard value that depends on the i-th training point's ground truth, causing overfitting. To fix this, we use the same strategy as Chen (2020) and instead aim to maximize

the modified "leave-one-out" log likelihood

$$\log \mathcal{L}^{\text{train}}(\theta) \triangleq \sum_{i=1}^{n}\sum_{\delta=1}^{m}\Big[\mathbb{1}\{\Delta_i=\delta\}\log \psi^{\neg i}_{\delta,\kappa(Y_i)}(X_i;\theta) - \sum_{\ell=1}^{\kappa(Y_i)} \psi^{\neg i}_{\delta,\ell}(X_i;\theta)\Big], \quad (17)$$

where

$$\psi^{\neg i}_{\delta,\ell}(x;\theta) \triangleq \frac{\sum_{j\neq i}\mathbb{1}\{\Delta_j=\delta, Y_j=t_\ell\}\exp(-\|f(x;\theta)-f(X_j;\theta)\|^2)}{\sum_{j\neq i}\mathbb{1}\{Y_j\geq t_\ell\}\exp(-\|f(x;\theta)-f(X_j;\theta)\|^2)}.$$

The numerator and denominator each sums over all training points except for the i-th training point (i.e., $\psi^{\neg i}_{\delta,\ell}(x;\theta)$ is a leave-one-out version of $\psi_{\delta,\ell}(x;\theta)$). Note that equation (17) also drops the constant term in equation (16) as it does not affect the optimization.

In practice, we minimize the loss $-\frac{1}{n}\log\mathcal{L}^{\text{train}}(\theta)$ using minibatch gradient descent. In our experiments later, we further add a second loss term corresponding to DeepHit's ranking loss (Lee et al., 2018), so that the overall loss function we use is the same as DeepHit. As the details of this ranking loss is not essential to our exposition, we defer it to Appendix C.

Neural net initialization We use the same scalable tree ensemble warm-start strategy as Chen (2024b) referred to as Tree ensemble Under a Neural Approximation (TUNA), where the tree ensemble we use to warm-start is the recently developed competing risks model SurvivalBoost (Alberge et al., 2025). We defer the details to Appendix D.

3.2. Overall DKAJ Framework

The previous section shows how we can use minibatch gradient descent to learn neural net parameters stored in θ. Effectively, once we learn these parameters, the neural net $f(\cdot;\theta)$ maps each data point to an embedding space. Following Chen (2024b), we now cluster in this embedding space and compute summary functions, where these are slightly different from Chen's since we are in a competing risks setup.

Training We train a DKAJ model as follows:

1. Use the training procedure in Section 3.1 to approximately solve $\widehat{\theta} \triangleq \arg\min_\theta -\frac{1}{n}\log\mathcal{L}^{\text{train}}(\theta)$.
2. Compute embedding vector $\widetilde{X}_i \triangleq f(X_i;\widehat{\theta})$ for $i\in[n]$.
3. Run an exemplar-based clustering algorithm on embedding vectors $\widetilde{X}_1,\ldots,\widetilde{X}_n$. For simplicity, we follow Chen (2024b) and use ε-net clustering, which works as follows for a user-specified $\varepsilon>0$:
   (a) Initialize the set of exemplars: $\mathcal{Q}\leftarrow\{1\}$ (training point 1 starts as the only exemplar).
   (b) For $i=2,\ldots,n$:
      i. Find the closest exemplar to $\widetilde{X}_i$: $j\leftarrow\arg\min_{q\in\mathcal{Q}}\|\widetilde{X}_i-\widetilde{X}_q\|$.
      ii. If $\|\widetilde{X}_i-\widetilde{X}_j\|\leq\varepsilon$: assign training point i to be in exemplar j's cluster. Otherwise: set $\mathcal{Q}\rightarrow\mathcal{Q}\cup\{i\}$ (make training point i a new exemplar).

   After clustering, let $\mathcal{C}_q\subseteq[n]$ denote the training points assigned to be in exemplar q's cluster.
4. Per exemplar $q\in\mathcal{Q}$, compute the following summary functions:

$$d^{\text{cluster}}_{\delta,\ell}(q) \triangleq \sum_{j\in\mathcal{C}_q}\mathbb{1}\{\Delta_j=\delta, Y_j=t_\ell\} \quad \text{for } \delta\in[m],\ell\in[L], \quad (18)$$

$$n^{\text{cluster}}_{\ell}(q) \triangleq \sum_{j\in\mathcal{C}_q}\mathbb{1}\{Y_j\geq t_\ell\} \quad \text{for } \ell\in[L]. \quad (19)$$

Prediction For any test feature vector x, prediction proceeds by using the following steps, where there is a hyperparameter $\tau>0$ for how close an exemplar has to be to the test point (in the embedding space) to contribute to the prediction for x:

1. Compute embedding vector $\widetilde{x}\rightarrow f(x;\widehat{\theta})$.
2. Find all exemplars in $\mathcal{Q}$ whose embedding vector is within Euclidean distance τ of $\widetilde{x}$. Denote the resulting set of close-enough exemplars as $\mathcal{Q}(x;\tau)$.
3. Compute weighted summary functions:

$$d^{\text{DKAJ}}_{\delta,\ell}(x) \triangleq \sum_{q\in\mathcal{Q}(x;\tau)} K(x,X_q)d^{\text{cluster}}_{\delta,\ell}(q) \quad \text{for } \delta\in[m],\ell\in[L], \quad (20)$$

$$n^{\text{DKAJ}}_{\ell}(x) \triangleq \sum_{q\in\mathcal{Q}(x;\tau)} K(x,X_q)n^{\text{cluster}}_{\ell}(q) \quad \text{for } \ell\in[L], \quad (21)$$

   where $K(x,x')=\exp(-\|f(x;\widehat{\theta})-f(x';\widehat{\theta})\|^2)$.
4. Finally, we predict all events' CIFs. To do this, we first compute the survival function estimate

$$\widehat{S}^{\text{DKAJ}}(t|x) \triangleq \prod_{\ell=1}^{L}\Big(1-\frac{\sum_{\delta=1}^{m}d^{\text{DKAJ}}_{\delta,\ell}(x)}{n^{\text{DKAJ}}_{\ell}(x)}\Big)^{\mathbb{1}\{t_\ell\leq t\}}.$$

   Then we compute

$$\widehat{F}^{\text{DKAJ}}_{\delta}(t|x) \triangleq \sum_{\ell=1}^{L}\frac{\mathbb{1}\{t_\ell\leq t\}d^{\text{DKAJ}}_{\delta,\ell}(x)}{n^{\text{DKAJ}}_{\ell}(x)}\widehat{S}^{\text{DKAJ}}(t_{\ell-1}|x).$$

   As a corner case, if the set of exemplars close enough to the test point is empty (i.e., $\mathcal{Q}(x;\tau)=\emptyset$), then simply predict the population-level Aalen-Johansen estimator from equation (6).

Special cases If $\varepsilon=0$ for ε-net clustering (so every training point is in its own cluster) and $\tau=\infty$ (so

that there's effectively no distance threshold), then the predicted CIF simplifies to be of the form:

$$\widehat{F}_{\delta}^{\mathrm{DKAJ}}(t|x) = \sum_{\ell=1}^{L} \frac{\mathbb{1}\{t_\ell \le t\} d_{\delta,\ell}(x)}{n_\ell(x)} \widehat{S}^{\mathrm{DKAJ}}(t_{\ell-1}|x),$$

where

$$\widehat{S}^{\mathrm{DKAJ}}(t|x) = \prod_{\ell=1}^{L} \left(1 - \frac{\sum_{\delta=1}^{m} d_{\delta,\ell}(x)}{n_\ell(x)}\right)^{\mathbb{1}\{t_\ell \le t\}}.$$

As a reminder, $d_{\delta,\ell}(x)$ and $n_\ell(x)$ are given in equations (12) and (13). When the training set size n is large, this predicted CIF can be expensive to compute since, in principle, for any test point x, we would compute $K(x, X_i)$ for all $i \in [n]$. If furthermore $K(x, x') = 1$ for all x, x', then the DKAJ estimator becomes the population-level AJ estimator (6) (as we see shortly, another way to set DKAJ hyperparameters also yields the population-level AJ estimator).

As a second special case, if there is only one critical event type (i.e., $m = 1$), then the overall DKAJ framework corresponds to survival kernets by Chen (2024b), where there would be no need to compute the CIF at the end (when there is only one critical event type, the survival function estimate $\widehat{S}^{\mathrm{DKAJ}}(t|x)$ would be sufficient as the only predicted output).

As a third special case, we point out that in step 2 of the prediction procedure, if only one exemplar is within distance τ of x in the embedding space (i.e., $|\mathcal{Q}(x;\tau)| = 1$), then the predicted CIF corresponds to the AJ estimator from equation (6) applied to only data points in that single exemplar's cluster.

As an extreme example of the third special case, if for ε-net clustering, we set $\varepsilon = \infty$ (so all training points are in the same cluster), then the predicted CIF becomes the population-level AJ estimator (6).

Model interpretation Any point with feature vector x is represented as a weighted combination of clusters. In particular, from looking at equations (20) and (21), the only exemplars that contribute to the prediction for x are the ones in the set $\mathcal{N}(x) \triangleq \{q \in \mathcal{Q}(x;\tau) : K(x, X_q) > 0\}$. Each exemplar q in $\mathcal{N}(x)$ has weight $K(x, X_q)$; in fact we can normalize the weights to sum to 1.[6] Thus, we know precisely which exemplars/clusters contribute to the prediction for x and with what weights.

From our discussion of special cases above, a key takeaway is that when using ε-net clustering, as $\varepsilon \to \infty$, the number of clusters decreases until there is a single cluster (and when $\varepsilon = 0$, every training point is its own cluster). A DKAJ model is easier to interpret if it has fewer, coarser clusters (corresponding to when ε is larger). In practice, one could tune ε based on an evaluation metric on the validation set.

After training a DKAJ model, it is possible to make visualizations to interpret any cluster as well as any data point. We defer showing these visualizations to the next section when we cover experiments (our visualizations depend on our experimental setup).

Optional summary function fine-tuning Survival kernets (Chen, 2024b) applies an optional final training step that treats $d_{\delta,\ell}^{\mathrm{cluster}}(q)$ and $n_\ell^{\mathrm{cluster}}(q)$ in equations (18) and (19) as trainable neural nets. We can fine-tune these summary functions to minimize the negative log likelihood loss. The prediction procedure remains the same. Extending Chen's strategy to handle competing risks is straightforward; for details, see Appendix E. For simplicity, in the main paper we do not show results using this optional step but we show results with this optional step in Appendix G.1.

4. Experiments

We now turn to numerical experiments, where our main goals are to show how well DKAJ works in practice (benchmarked against classical and state-of-the-art baselines on standard datasets) and to also illustrate how to make visualizations to interpret a trained DKAJ model.

Datasets We use the following four standard competing risks datasets (see Table 1 for summary statistics), each with two critical event types (we refer to one as "primary" and the other as "competing"):

- PBC (Fleming and Harrington, 1991): a cohort of 1,945 patients with primary biliary cirrhosis (PBC) described by 15 features. The primary event is death, and the competing event is liver transplant.
- Framingham (Kannel and McGee, 1979): a longitudinal study of 4,434 participants with 18 baseline features aimed at identifying cardiovascular disease (CVD) risk factors. We restrict our analysis to the earliest available feature measurements per patient to make the dataset tabular and not longitudinal. The primary event is death from CVD, with death from other causes as the competing event.
- SEER: a subset of 24,907 female breast cancer patients from the SEER[7] registry, restricted to diagnoses in the year 2010, and described by 22 demo-

6. For each x, we can normalize the weights $K(x, X_q)$ for $q \in \mathcal{N}(x)$ to sum to 1 since the normalization factor would cancel out when we divide equation (20) by (21) in the calculation of $\widehat{F}_{\delta}^{\mathrm{DKAJ}}(t|x)$ as well as $\widehat{S}^{\mathrm{DKAJ}}(t|x)$).

7. https://seer.cancer.gov/

Table 1: Dataset summary statistics. Every dataset has two competing event types (primary vs competing).

Dataset	Number of Data Points	Features	Primary Event	Competing Event	Censoring Rate
PBC	1,945	15	Death (37.3%)	Transplant (7.6%)	55.2%
Framingham	4,434	18	CVD Death (26.1%)	Non-CVD Death (17.7%)	56.2%
SEER	24,907	22	BC (16.7%)	CVD (4.4%)	78.9%
Synthetic	30,000	12	Event 1 (25.3%)	Event 2 (24.7%)	50.0%

graphic and disease-related features. The primary event is breast cancer (BC) mortality, with cardiovascular disease (CVD) as the competing event.

- Synthetic (Lee et al., 2018): a simulated dataset of 30,000 individuals with 12 features, constructed with two hypothetical competing events and subject to 50% random censoring.

Baselines We use the following baselines:

- Classical competing risks models: the Fine and Gray model (abbreviated as Fine-Gray) (Fine and Gray, 1999), cause-specific Cox (cs-Cox) (Prentice et al., 1978), and random survival forest with competing risks (RSF-CR) (Ishwaran et al., 2014).
- Deep learning–based methods: DeepHit (Lee et al., 2018), Deep Survival Machines (DSM) (Nagpal et al., 2021), and Neural Fine-Gray (NeuralFG) (Jeanselme et al., 2023).
- SurvivalBoost (Alberge et al., 2025), a recently developed gradient boosted tree ensemble model for competing risks.

Aside from Fine-Gray and cs-Cox, the other baselines are not designed to be inherently interpretable.[8]

Evaluation metrics Model performance is assessed using time-dependent concordance index (C^{td}) (Antolini et al., 2005) and integrated Brier score (IBS) for competing risks. Formal definitions and implementation details are provided in Appendix F.2.

Experimental setup For each dataset, we randomly split the data into 70% training and 30% testing subsets. Within the training subset, we further split the data into 80% "proper" training and 20% validation subsets. Models are trained on the proper training set under varying hyperparameter configurations, and the validation set is used for model selection. Final performance is reported on the held-out test set. For every dataset and method, this procedure is repeated 10 times with different random splits. Details of the hyperparameter search space and optimization strategy are provided in Appendix F.3.

Table 2: Test set C^{td} (mean ± std. dev. across 10 random splits) for primary and competing events (best score in **bold**, 2nd best in blue).

Dataset	Method	Primary	Competing
PBC	Fine-Gray	0.8212±0.0119	0.8769±0.0203
	cs-Cox	0.8295±0.0102	0.9094±0.0132
	RSF-CR	0.8584±0.0089	0.8746±0.0163
	DeepHit	0.8407±0.0131	0.9060±0.0130
	DSM	0.8319±0.0115	0.9039±0.0189
	NeuralFG	0.8363±0.0150	0.9120±0.0149
	SurvivalBoost	**0.8719±0.0107**	**0.9360±0.0114**
	DKAJ	0.8413±0.0105	0.9039±0.0170
Framingham	Fine-Gray	0.7733±0.0106	0.7144±0.0182
	cs-Cox	**0.7753±0.0105**	**0.7160±0.0173**
	RSF-CR	0.7720±0.0098	0.6982±0.0119
	DeepHit	0.7423±0.0198	0.6957±0.0195
	DSM	0.7664±0.0170	0.7095±0.0184
	NeuralFG	0.6833±0.0619	0.6978±0.0287
	SurvivalBoost	0.7645±0.0152	0.7031±0.0156
	DKAJ	0.7677±0.0138	0.7076±0.0209
SEER	Fine-Gray	0.8151±0.0021	0.8478±0.0099
	cs-Cox	0.7630±0.0085	0.8415±0.0132
	RSF-CR	0.8350±0.0030	0.8301±0.0137
	DeepHit	0.8239±0.0028	0.8548±0.0120
	DSM	0.7807±0.0088	0.8422±0.0119
	NeuralFG	0.7743±0.0090	0.8362±0.0138
	SurvivalBoost	**0.8418±0.0038**	0.8583±0.0081
	DKAJ	0.8333±0.0034	**0.8598±0.0072**
Synthetic	Fine-Gray	0.5823±0.0051	0.5917±0.0071
	cs-Cox	0.5809±0.0051	0.5902±0.0073
	RSF-CR	0.7224±0.0071	0.7203±0.0043
	DeepHit	0.7401±0.0067	0.7437±0.0043
	DSM	0.7280±0.0055	0.7320±0.0042
	NeuralFG	**0.7481±0.0073**	**0.7513±0.0045**
	SurvivalBoost	0.7160±0.0083	0.7190±0.0039
	DKAJ	0.7384±0.0067	0.7435±0.0041

Model performance Table 2 reports the test set C^{td} values (mean ± standard deviation) across 10 different experimental repeats. For each run, hyperparameters are selected based on the model achieving the best validation C^{td}. We also performed model selection using the validation IBS as the criterion. The corresponding results for both C^{td} and IBS are pro-

8. RSF-CR is typically used with a large number of trees each with many leaves so that interpretation is not straightforward. Separately, when we talk about a model being "inherently interpretable", we mean that the model's prediction itself comes with an explanation that does not rely on general post-hoc explanation tools such as Shapley-value-based approaches (e.g., Shapley 1953; Lundberg and Lee 2017) or feature importance by permutation (e.g., Breiman 2001; Ishwaran et al. 2014; Fisher et al. 2019).

vided in Appendix G.1 under both circumstances.[9] Model performance with the optional summary function fine-tuning is also reported in Appendix G.1.

We highlight two key takeaways: first, no single method consistently outperforms the others across all datasets and event types. Second, our proposed DKAJ estimator achieves performance that is competitive with state-of-the-art baselines. In more detail, we conduct paired Wilcoxon signed-rank tests across all methods (Appendix G.2), and find that DKAJ achieves significantly lower IBS and competitive or higher C^{td} relative to most baselines, while differences among the top-performing models are generally not significant. DKAJ is also amenable to model interpretation, which we explore momentarily in terms of cluster-level and individual-level visualizations. Again, the only baselines we compared against that are interpretable are Fine-Gray and cs-Cox, both of which provide a population-level notion of interpretability (e.g., global hazard ratios).

Ablation study We remove two components of DKAJ to assess their impact on model performance: (1) the leave-one-out (LOO) likelihood in equation (17), replaced by the standard likelihood in equation (16); and (2) the TUNA warm start, replaced by PyTorch's default initialization. As shown in Table G.7, for all three real-world datasets (PBC, Framingham, and SEER), removing either component results in worse performance in most cases. Details are provided in Appendix G.4.

Model scalability We assess model scalability in two manners. First, we vary the training set size from 10% to 70% on the synthetic dataset, with the remaining 30% held out for testing (see Figure 1 for results on Event 1). Across all models, the C^{td} improves with larger training sets and stabilizes once sufficient data are available. DKAJ exhibits consistent gains without signs of overfitting or instability, maintaining performance comparable to other neural and ensemble baselines across all data regimes. Results for Event 2 show similar trends (Appendix G.5). We separately show that DKAJ has computation cost (wall-clock training time and model size) comparable to recently developed methods (Appendix G.6).

9. The choice of validation criterion has a substantial impact: selecting models based on IBS generally yields lower C^{td}, and conversely, selecting based on C^{td} tends to yield worse IBS. This issue is well-known in survival analysis literature (e.g., Lillelund et al. 2025), which motivates our decision to report both metrics under both model selection strategies.

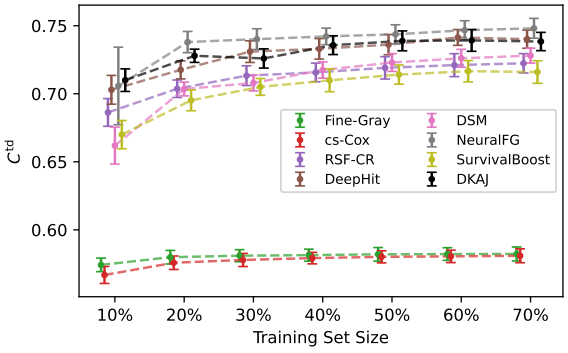

Figure 1: C^{td} on the synthetic dataset (Event 1) as training set size varies; 30% held-out test

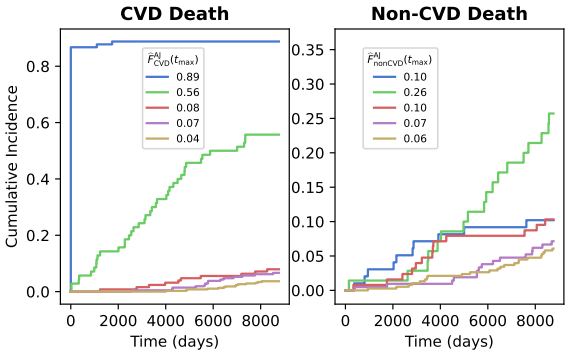

Figure 2: (Framingham) CIFs for the largest 5 clusters (we then sort these 5 clusters in decreasing order by the estimated probability of CVD death happening within the maximum observed time). Clusters correspond across the two plots in this figure as well as in Figure 3 (e.g., the blue cluster is the same cluster across these visualizations).

Cluster-level visualizations For a trained DKAJ model, we can visualize the (a) CIFs of any cluster, and also (b) a heatmap showing the distributions of features of any cluster. We illustrate these two visualizations (a) and (b) in Figures 2 and 3 respectively for the Framingham dataset, where we restrict our attention to the five largest clusters for simplicity (largest meaning the most number of training points assigned to them). We then sort these clusters by the estimated probability of experiencing the primary event (CVD death) earliest. Specifically, clusters can be ranked by the CIF of the primary event evaluated at the maximum observed training event time $t_{\max}$.

In more detail, Figure 2 shows the CIFs for the 5 largest clusters, ordered by the risk of CVD death (i.e., CIF of CVD death evaluated at $t_{\max}$); this risk is also displayed per cluster in the legend. The corresponding clusters' feature distributions are shown

in Figure 3. For example, the blue cluster (estimated risk of CVD death = 0.89) exhibits high CVD risk and consists of patients with a history of hypertension (PREVHYP), coronary heart disease (PREVCHD), angina pectoris (PREVAP), current smoking (CURSMOKE), and older age. The green cluster (estimated risk of CVD death = 0.56) also shows high CVD mortality, but in addition displays an elevated risk of death from other causes; this cluster is characterized by higher blood pressure (SYSBP, DIABP) and BMI, but a lower prevalence of prior coronary heart disease, suggesting a different clinical profile. In contrast, the other three clusters have substantially lower cumulative incidence of both CVD and non-CVD death. Such visualizations highlight similarities and differences between high-risk groups, and illustrate how the model's clusters align with meaningful clinical subpopulations.

A kernel-similarity heatmap for Framingham (50 largest clusters) is provided in Appendix H, Figure H.7, showing a clear block-diagonal pattern consistent with coherent clusters of varying sizes.

Visualizations for the 5 largest clusters in PBC and SEER datasets are provided in Appendix H. We omit visualizations for the synthetic dataset as this dataset does not have a clinical interpretation to begin with.

Individual-level visualizations Per test subject, we can examine the subject's predicted CIF curves directly. In addition, the cluster weights assigned by the DKAJ model provide an interpretable decomposition: by identifying which clusters contribute most strongly to a prediction, we can inspect the defining characteristics of those clusters and thus better understand which clinical factors the model considers most relevant for that individual. Furthermore, conditional median times to each event, given that event occurs earliest, can be reported. Examples of individual-level visualizations are in Appendix H.

5. Discussion

We have proposed a new flexible and interpretable competing risks model DKAJ that is competitive with various state-of-the-art baselines. Some possible extensions include using a proper scoring rule (such as the one by Alberge et al. (2025)) as a training loss, figuring out better ways to tune hyperparameters especially to account for ease of model interpretation, trying other competing risks evaluation metrics (e.g., PDI (Ding et al., 2021)), and generalizing beyond competing risks to multistate processes.

We would like to provide some commentary regarding DKAJ's kernel function formulation and choice of

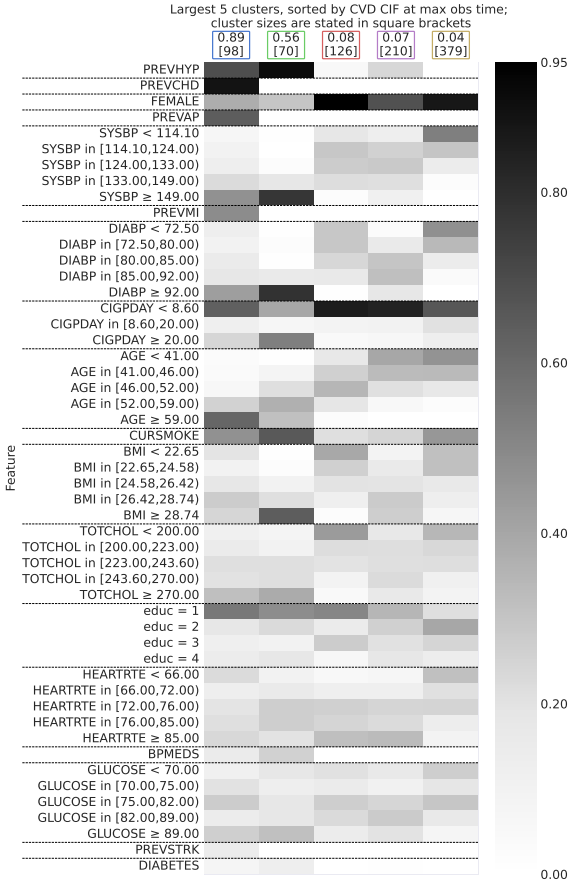

Figure 3: (Framingham) Feature heatmap summarizing distributions of variables in the same 5 clusters as in Figure 2. Darker shades mean higher feature values or frequencies.

clustering method. For simplicity, we defined the kernel function to resemble a Gaussian kernel but other options are possible (e.g., Laplacian, Cauchy). Moreover, it is possible to have the kernel function be time-dependent (this is done by SurvivalBoost; see Appendix D), or to learn a separate kernel function per critical event type—although these make model interpretation more involved. For clustering, we follow Chen (2024b) and use ε-net clustering. Chen used ε-net clustering as it enables establishing a theoretical error bound for survival kernets. We suspect that establishing a similar such guarantee for DKAJ is also possible but defer this to future work. In practice, other exemplar-based clustering methods (e.g., affinity propagation, k-medoids) can also be used. Technically, using a non-exemplar-based clustering method like k-means can be done but the cluster centers then would no longer correspond to actual training patients, which can complicate model interpretation.

Acknowledgments

This work was supported by NSF CAREER award #2047981. The authors thank Yu Cheng, Ying Ding, and the anonymous reviewers for helpful feedback.

References

Odd O Aalen and Søren Johansen. An empirical transition matrix for non-homogeneous markov chains based on censored observations. *Scandinavian Journal of Statistics*, pages 141–150, 1978.

Julie Alberge, Vincent Maladiere, Olivier Grisel, Judith Abécassis, and Gaël Varoquaux. Survival models: Proper scoring rule and stochastic optimization with competing risks. In *International Conference on Artificial Intelligence and Statistics*, 2025.

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

Martin Bladt and Christian Furrer. Conditional Aalen–Johansen estimation. *Scandinavian Journal of Statistics*, 52(2):873–902, 2025.

Leo Breiman. Random forests. *Machine learning*, 45: 5–32, 2001.

Norman Breslow. Discussion of the paper by D R Cox (1972). *Journal of the Royal Statistical Society, Series B*. 34(2):216–217, 1972.

George H Chen. Deep kernel survival analysis and subject-specific survival time prediction intervals. In *Machine Learning for Healthcare Conference*, pages 537–565. PMLR, 2020.

George H Chen. An introduction to deep survival analysis models for predicting time-to-event outcomes. *Foundations and Trends in Machine Learning*, 17(6):921–1100, 2024a.

George H Chen. Survival kernets: Scalable and interpretable deep kernel survival analysis with an accuracy guarantee. *Journal of Machine Learning Research*, 25(40):1–78, 2024b.

Richard J Cook and Jerald F Lawless. *Multistate Models for the Analysis of Life History Data*. Chapman and Hall/CRC, 2018.

David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B*, 34(2):187–202, 1972.

Dominic Danks and Christopher Yau. Derivative-based neural modelling of cumulative distribution functions for survival analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 7240–7256. PMLR, 2022.

Cameron Davidson-Pilon. lifelines: survival analysis in Python. *Journal of Open Source Software*, 4(40): 1317, 2019.

Maomao Ding, Jing Ning, and Ruosha Li. Evaluation of competing risks prediction models using polytomous discrimination index. *Canadian Journal of Statistics*, 49(3):731–753, 2021.

Jessie K Edwards, Laura L Hester, Mugdha Gokhale, and Catherine R Lesko. Methodologic issues when estimating risks in pharmacoepidemiology. *Current Epidemiology Reports*, 3(4):285–296, 2016.

Jason P Fine and Robert J Gray. A Proportional Hazards Model for the Subdistribution of a Competing Risk. *Journal of the American Statistical Association*, 94(446):496–509, 1999.

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

Thomas R Fleming and David P Harrington. *Counting Processes and Survival Analysis*. John Wiley & Sons, 1991.

Thomas A Gerds and Michael W Kattan. *Medical risk prediction models: with ties to machine learning*. Chapman and Hall/CRC, 2021.

Hemant Ishwaran, Thomas A Gerds, Udaya B Kogalur, Richard D Moore, Stephen J Gange, and Bryan M Lau. Random survival forests for competing risks. *Biostatistics*, 15(4):757–773, 2014.

Vincent Jeanselme, Chang Ho Yoon, Brian Tom, and Jessica Barrett. Neural Fine-Gray: Monotonic neural networks for competing risks. In *Conference on Health, Inference, and Learning*, pages 379–392. PMLR, 2023.

John D Kalbfleisch and Ross L Prentice. *The Statistical Analysis of Failure Time Data (2nd ed.)*. John Wiley & Sons, 2002.

William B Kannel and Daniel L McGee. Diabetes and cardiovascular disease: the Framingham study. *JAMA*, 241(19):2035–2038, 1979.

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

Håvard Kvamme and Ørnulf Borgan. Continuous and discrete-time survival prediction with neural networks. *Lifetime data analysis*, 27(4):710–736, 2021.

Havard 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.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar. DeepHit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Christian Marius Lillelund, Shi-ang Qi, Russell Greiner, and Christian Fischer Pedersen. Stop chasing the c-index: This is how we should evaluate our survival models. *arXiv preprint arXiv:2506.02075*, 2025.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017.

Chirag Nagpal, Xinyu Li, and Artur Dubrawski. Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics*, 25(8):3163–3175, 2021.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ross L Prentice, John D Kalbfleisch, Arthur V Peterson Jr, Nancy Flournoy, Vernon T Farewell, and Norman E Breslow. The analysis of failure times in the presence of competing risks. *Biometrics*, pages 541–554, 1978.

David Rindt, Robert Hu, David Steinsaltz, and Dino Sejdinovic. Survival regression with proper scoring rules and monotonic neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1190–1205. PMLR, 2022.

L S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 1953.

Appendix A. Details on Proposition 1: Proof with Explanation of Interpolation

Let $0 < t_1 < t_2 < \cdots < t_L$ denote the unique times in which any critical event occurs in the training data. We also define $t_0 \triangleq 0$. Collectively, $t_0 < t_1 < \cdots < t_L$ forms our discrete time grid. For an observed time Y_i that corresponds to an uncensored data point (i.e., $\Delta_i \neq 0$), let $\kappa(Y_i) \in [L]$ denote the time index corresponding to Y_i. Following the preprocessing that we referenced in footnote 5, we take a censored data point's observed time to be the preceding uncensored event time. Thus, if $\Delta_i = 0$, then

$$\kappa(Y_i) = \max\big\{\ell \in \{0, 1, \ldots, L\} \text{ s.t. } t_\ell < Y_i\big\}.$$

We use the parameterization from equation (8), which we reproduce below for convenience:

$$\lambda_\delta(t|x) = \begin{cases} \dfrac{\phi_{\delta,\ell}}{t_\ell - t_{\ell-1}} & \text{if } t \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(equation (8), reproduced)}$$

As a reminder, this parameterization does not depend on the input feature vector x, so that we can write $\lambda_\delta(t|x) \triangleq \lambda_\delta(t)$. We plug this parameterization into the log of the likelihood $\mathcal{L}$ in equation (3) to get

$$\begin{aligned}
&\log \mathcal{L} \\
&= \sum_{\delta=1}^{m}\sum_{i=1}^{n}\left[\mathbb{1}\{\Delta_i = \delta\}\log\lambda_\delta(Y_i|X_i) - \int_0^{Y_i}\lambda_\delta(u|X_i)\mathrm{d}u\right] \\
&= \sum_{\delta=1}^{m}\sum_{i=1}^{n}\left[\mathbb{1}\{\Delta_i = \delta\}\log\phi_{\delta,\kappa(Y_i)} \right. \\
&\qquad\qquad \left. - \sum_{\ell=1}^{\kappa(Y_i)}\frac{\phi_{\delta,\ell}}{t_\ell - t_{\ell-1}}(t_\ell - t_{\ell-1})\right] \\
&= \sum_{\delta=1}^{m}\sum_{i=1}^{n}\left[\mathbb{1}\{\Delta_i = \delta\}\log\phi_{\delta,\kappa(Y_i)} - \sum_{\ell=1}^{\kappa(Y_i)}\phi_{\delta,\ell}\right] \\
&= \sum_{\delta=1}^{m}\sum_{i=1}^{n}\sum_{\ell=1}^{\kappa(Y_i)}\big[\mathbb{1}\{\kappa(Y_i) = \ell, \Delta_i = \delta\}\log\phi_{\delta,\ell} - \phi_{\delta,\ell}\big] \\
&= \sum_{\delta=1}^{m}\sum_{\ell=1}^{L}\sum_{i=1}^{n}\big[\mathbb{1}\{\kappa(Y_i) = \ell, \Delta_i = \delta\}\log\phi_{\delta,\ell} \\
&\qquad\qquad - \mathbb{1}\{\kappa(Y_i) \geq \ell\}\phi_{\delta,\ell}\big]. \qquad (22)
\end{aligned}$$

Note that the condition $\kappa(Y_i) = \ell$ is equivalent to $Y_i = t_\ell$. Similarly, the condition $\kappa(Y_i) \geq \ell$ is equivalent to $Y_i \geq t_\ell$. Thus,

$$\begin{aligned}
\log\mathcal{L} = \sum_{\delta=1}^{m}\sum_{\ell=1}^{L}\sum_{i=1}^{n}\big[&\mathbb{1}\{Y_i = t_\ell, \Delta_i = \delta\}\log\phi_{\delta,\ell} \\
&- \mathbb{1}\{Y_i \geq t_\ell\}\phi_{\delta,\ell}\big].
\end{aligned}$$

Setting the derivative with respect to $\phi_{\delta,\ell}$ to 0, we get

$$\begin{aligned}
0 &= \Big[\frac{\partial\log\mathcal{L}}{\partial\phi_{\delta,\ell}}\Big]_{\phi_{\delta,\ell}=\widehat{\phi}_{\delta,\ell}} \\
&= \frac{\sum_{i=1}^{n}\mathbb{1}\{Y_i = t_\ell, \Delta_i = \delta\}}{\widehat{\phi}_{\delta,\ell}} - \sum_{i=1}^{n}\mathbb{1}\{Y_i \geq t_\ell\}.
\end{aligned}$$

Rearranging terms yields

$$\widehat{\phi}_{\delta,\ell} = \frac{\sum_{i=1}^{n}\mathbb{1}\{Y_i = t_\ell, \Delta_i = \delta\}}{\sum_{i=1}^{n}\mathbb{1}\{Y_i \geq t_\ell\}} = \frac{d_{\delta,\ell}}{n_\ell}.$$

One can verify that $[\frac{\partial^2\log\mathcal{L}}{\partial\phi_{\delta,\ell}^2}]_{\phi_{\delta,\ell}=\widehat{\phi}_{\delta,\ell}} < 0$ so that $\widehat{\phi}_{\delta,\ell}$ is a maximum (in fact, it's the global maximum since $\log\mathcal{L}$ is concave). Moreover, $\widehat{\phi}_{\delta,\ell}$ is clearly nonnegative (although we did not explicitly enforce this constraint in the optimization, it is a constraint that does need to be satisfied). Plugging in $\widehat{\phi}_{\delta,\ell}$ in place of $\phi_{\delta,\ell}$ in equation (8), we get that the maximum likelihood estimate of the event-specific hazard function is

$$\widehat{\lambda}_\delta(t) = \begin{cases} \dfrac{d_{\delta,\ell}}{(t_\ell - t_{\ell-1})n_\ell} & \text{if } t_\ell \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ 0 & \text{otherwise.} \end{cases}$$

We can then integrate to get the *event-specific cumulative hazard function*:

$$\begin{aligned}
\widehat{\Lambda}_\delta(t_\ell) &\triangleq \int_0^{t_\ell}\widehat{\lambda}_\delta(u)\mathrm{d}u \\
&= \sum_{a=1}^{\ell}\frac{d_{\delta,a}}{(t_a - t_{a-1})n_a}(t_a - t_{a-1}) \\
&= \sum_{a=1}^{\ell}\frac{d_{\delta,a}}{n_a}.
\end{aligned}$$

Note that the equation above is evaluating $\widehat{\Lambda}_\delta(t)$ specifically at time points along the discrete time grid t_ℓ for $\ell \in [L]$. When we evaluate at times in between these discrete grid points, the integration does constant-hazard interpolation. In particular, for any time $t \in (t_{\ell-1}, t_\ell)$ (with $\ell \in [L]$), we have

$$\widehat{\Lambda}_\delta(t) = \sum_{a=1}^{\ell-1}\frac{d_{\delta,a}}{n_a} + \frac{d_{\delta,\ell}}{(t_\ell - t_{\ell-1})n_\ell}(t - t_{\ell-1}),$$

with the convention that when $\ell = 1$, the first term on the right-hand side is 0.

However, at this point, to obtain the actual commonly used version of the Aalen-Johansen estimator stated in equation (6), instead of doing this constant-hazard interpolation, we do a "forward-filling" interpolation. Namely, we approximate the event-specific cumulative hazard function with

$$\widehat{\Lambda}_\delta^{\text{forward-fill}}(t) \triangleq \begin{cases} 0 & \text{if } t = 0 \\ \sum_{a=1}^{\ell} \frac{d_{\delta,a}}{n_a} & \text{if } t \in (t_{\ell-1}, t_\ell] \text{ for } \ell \in [L], \\ \sum_{a=1}^{L} \frac{d_{\delta,a}}{n_a} & \text{if } t > t_L. \end{cases}$$

A similar derivation can be used to show that the classical Kaplan-Meier estimator is a forward-filled interpolated version of a maximum likelihood estimator (for a derivation of this result, see Example 2.6 and Section 2.A.3 of Chen (2024a); note that we basically reduce the setup to a standard survival analysis setup where there is a single critical event, corresponding to any of the m critical events happening).

We now finally use the main idea of the Aalen-Johansen estimator (Aalen and Johansen, 1978). For reference, the general Aalen-Johansen estimator for multistate survival analysis is given by the matrix product in equation (3.17) of Cook and Lawless (2018). We instead use the special case given by equation (4.3) in Cook and Lawless's book, which states that, using Riemann-Stieltjes integral notation, we can estimate $F_\delta^{\text{pop}}(t) = \mathbb{P}(T \le t, \Delta^* = \delta)$ with

$$\int_0^t \widehat{S}(u^-) \mathrm{d}\widehat{\Lambda}_\delta(u),$$

where $\widehat{S}$ is any estimator of the survival function $S(t) = \mathbb{P}(T > t)$, and $\widehat{\Lambda}_\delta$ is any estimator of the event-specific cumulative hazard function Λ_δ (these functions are all at the population level and are not conditioned on a feature vector); "$\widehat{S}(u^-)$" refers to evaluating $\widehat{S}$ at the time immediately before u. When we plug in $\widehat{S}^{\text{KM}}$ (from equation (7)) for $\widehat{S}$ and $\widehat{\Lambda}_\delta^{\text{forward-fill}}$ for $\widehat{\Lambda}_\delta$, this Riemann-Stieltjes integral can be evaluated in closed-form and is precisely equal to

$$\widehat{F}_\delta^{\text{AJ}}(t) \triangleq \sum_{\ell=1}^{L} \widehat{S}^{\text{KM}}(t_{\ell-1}) \frac{\mathbb{1}\{t_\ell \le t\} d_{\delta,\ell}}{n_\ell}.$$

This completes the proof and also indicates the interpolation used (forward-filling interpolation). ■

Appendix B. DKAJ Log Likelihood Calculation

The derivation is similar to equation (22). Using equations (3), (10), and (15), we have

$$\begin{aligned}
&\log \mathcal{L} \\
&= \sum_{\delta=1}^{m} \sum_{i=1}^{n} \left[\mathbb{1}\{\Delta_i = \delta\} \log \lambda_\delta(Y_i|X_i) - \int_0^{Y_i} \lambda_\delta(u|X_i) \mathrm{d}u \right] \\
&= \sum_{\delta=1}^{m} \sum_{i=1}^{n} \left[\mathbb{1}\{\Delta_i = \delta\} \log \frac{\psi_{\delta,\kappa(Y_i)}(X_i;\theta)}{t_{\kappa(Y_i)} - t_{\kappa(Y_i)-1}} \right. \\
&\qquad \left. - \sum_{\ell=1}^{\kappa(Y_i)} \frac{\psi_{\delta,\ell}(X_i;\theta)}{t_\ell - t_{\ell-1}} (t_\ell - t_{\ell-1}) \right] \\
&= \sum_{\delta=1}^{m} \sum_{i=1}^{n} \left[\mathbb{1}\{\Delta_i = \delta\} \log \frac{\psi_{\delta,\kappa(Y_i)}(X_i;\theta)}{t_{\kappa(Y_i)} - t_{\kappa(Y_i)-1}} \right. \\
&\qquad \left. - \sum_{\ell=1}^{\kappa(Y_i)} \psi_{\delta,\ell}(X_i;\theta) \right] \\
&= \sum_{\delta=1}^{m} \sum_{i=1}^{n} \left[\mathbb{1}\{\Delta_i = \delta\} \log \psi_{\delta,\kappa(Y_i)}(X_i;\theta) \right. \\
&\qquad \left. - \sum_{\ell=1}^{\kappa(Y_i)} \psi_{\delta,\ell}(X_i;\theta) \right] \\
&\quad \underbrace{- \sum_{\delta=1}^{m} \sum_{i=1}^{n} \mathbb{1}\{\Delta_i = \delta\} \log(t_{\kappa(Y_i)} - t_{\kappa(Y_i)-1})}_{\text{constant w.r.t. neural net parameters } \theta}.
\end{aligned}$$

Lastly, to emphasize the dependence of the log likelihood on θ, we use the notation "$\log \mathcal{L}(\theta)$" instead of just "$\log \mathcal{L}$".

Appendix C. DeepHit Loss

DeepHit (Lee et al., 2018) uses both a negative log likelihood loss and a ranking loss. The negative log likelihood loss is precisely $\mathbb{L}_{\text{NLL}} = -\frac{1}{n} \log \mathcal{L}^{\text{train}}$, where $\log \mathcal{L}^{\text{train}}$ is given in equation (17). For hyperparameter $\sigma > 0$, the DeepHit ranking loss can be stated as

$$\mathbb{L}_{\text{ranking}} \triangleq \frac{1}{n^2} \sum_{\delta=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{1}\{\Delta_i = \delta, Y_i < Y_j\} \Xi_{\delta,i,j}$$

where

$$\Xi_{\delta,i,j} \triangleq \exp\left(\frac{\widehat{F}_\delta^{\text{DKAJ}}(Y_i|X_j) - \widehat{F}_\delta^{\text{DKAJ}}(Y_i|X_i)}{\sigma} \right).$$

Note that the scale factor we use for the ranking loss is different from the original DeepHit paper but cor-

responds to the implementation in the now-standard `pycox` software package by Kvamme et al. (2019). The overall DeepHit loss, following the `pycox` implementation, is

$$\mathbb{L}_{\text{DeepHit}} \triangleq \alpha \mathbb{L}_{\text{NLL}} + (1-\alpha)\mathbb{L}_{\text{ranking}},$$

where $\alpha \in [0,1]$ is a hyperparameter.

Appendix D. TUNA Warm-up with SurvivalBoost

For warm-starting neural net training for large datasets, the survival kernets paper (Chen, 2024b) found that using a decision tree ensemble method that scales to large datasets is very helpful in practice (in a warm-start procedure called Tree ensemble Under a Neural Approximation (TUNA)). We wanted to adopt this TUNA warm-start strategy but the tree ensemble model that survival kernets used (XGBoost) does not, to the best of our knowledge, support competing risks. In fact, the only competing risks decision tree ensemble model we know of that scales to large datasets was only recently developed (SurvivalBoost by Alberge et al. (2025)). Our method uses SurvivalBoost to warm-start neural net training and not XGBoost. We point out two implementation details needed to get the TUNA warm-start with SurvivalBoost to work:

- First, the warm-start procedure needs to know predicted leaf IDs per decision tree in the tree ensemble. At the time of writing, this functionality is not provided by SurvivalBoost, so we coded this part ourselves, building off the SurvivalBoost codebase.
- TUNA relies on approximating the kernel function of the base tree ensemble model (for two data points, the kernel function value is the fraction of trees that place the two data points in the same leaf). However, for SurvivalBoost, whether two data points are in the same leaf for a specific tree is time-dependent (i.e., at two different evaluation times t_1 and t_2, it is possible that two data points are considered to be in the same leaf at time t_1 but not at time t_2). Under the hood, SurvivalBoost treats time as an additional feature (i.e., time is appended to the feature vector per data point) when training a tree ensemble. This complication did not arise in the original survival kernets paper when it used XGBoost to warm-start (the XGBoost survival models don't have this time-dependent behavior as to which data points are considered to be in the same leaf). To deal with this, we randomly sample 10 evaluation times per training patient during the warm-start procedure.

Appendix E. Summary Function Fine-Tuning

We now treat $d_{\delta,\ell}^{\text{cluster}}(q)$ and $n_{\ell}^{\text{cluster}}(q)$ as neural nets initialized to the values specified in equations (18) and (19). Specifically, we set

$$d_{\delta,\ell}^{\text{cluster}}(q) = \exp(\gamma_{q,\delta,\ell}) + \exp(\gamma_{\delta,\ell}^{\text{baseline}}),$$

where $\gamma_{q,\delta,\ell} \in \mathbb{R}$ and $\gamma_{\delta,\ell}^{\text{baseline}} \in \mathbb{R}$ are unconstrained neural net parameters; we collect these particular neural net parameters all into the variable γ. This means that γ consists of:

- $\gamma_{q,\delta,\ell}$ for $q \in \mathcal{Q}$, $\delta \in [m]$, and $\ell \in [L]$,
- $\gamma_{\delta,\ell}^{\text{baseline}}$ for $\delta \in [m]$, and $\ell \in [L]$.

We initialize each $\gamma_{\delta,\ell}^{\text{baseline}}$ to be a big negative value (e.g., $\gamma_{\delta,\ell}^{\text{baseline}} = \log(10^{-12}) = -27.631\ldots$), so that $\exp(\gamma_{\ell}^{\text{baseline}}) = 10^{-12}$. Meanwhile, we initialize each neural net parameter

$$\gamma_{q,\delta,\ell} = \log\Big(\sum_{j \in \mathcal{C}_q} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\}\Big).$$

Thus, at these initial values, $d_{\delta,\ell}^{\text{cluster}}(q)$ is approximately equal to $\sum_{j\in\mathcal{C}_q} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\}$ (corresponding to equation (18)).

Next, we explain how we model $n_{\ell}^{\text{cluster}}(q)$. Importantly, we do not parameterize this function directly. Instead, we define a new summary function corresponding to a pseudocount for how many points are censored at time ℓ for cluster q. Specifically, we define

$$c_{\ell}^{\text{cluster}}(q) = \exp(\omega_{q,\ell}) + \exp(\omega_{\ell}^{\text{baseline}}),$$

where $\omega_{q,\ell} \in \mathbb{R}$ and $\omega_{\ell}^{\text{baseline}} \in \mathbb{R}$ are unconstrained neural net parameters; we collect these parameters into the variable ω. This means that ω consists of:

- $\omega_{q,\ell}$ for $q \in \mathcal{Q}$ and $\ell \in [L]$,
- $\omega_{\ell}^{\text{baseline}}$ for $\ell \in [L]$.

We have the recurrence relation

$$n_{\ell}^{\text{cluster}}(q) = \sum_{\delta=1}^{m} d_{\delta,\ell}^{\text{cluster}}(q) + c_{\ell}^{\text{cluster}}(q) + n_{\ell+1}^{\text{cluster}}(q),$$

where $n_{L+1}^{\text{cluster}}(q) \triangleq 0$. Thus, we would like to initialize neural net parameters $\omega_{q,\ell}$ and $\omega_{\ell}^{\text{baseline}}$ as to

(approximately) satisfy

$$\begin{aligned} c_\ell^{\text{cluster}}(q) &= n_\ell^{\text{cluster}}(q) - n_{\ell+1}^{\text{cluster}}(q) - \sum_{\delta=1}^{m} d_{\delta,\ell}^{\text{cluster}}(q) \\ &= \sum_{j \in \mathcal{C}_q} \mathbb{1}\{Y_j \geq t_\ell\} - \sum_{j \in \mathcal{C}_q} \mathbb{1}\{Y_j \geq t_{\ell+1}\} \\ &\quad - \sum_{\delta=1}^{m} \sum_{j \in \mathcal{C}_q} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\}, \end{aligned}$$

where the second equality is from plugging in equations (18) and (19). It suffices to set $\omega_\ell^{\text{baseline}}$ to again be a large negative value, e.g., $\omega_\ell^{\text{baseline}} = \log(10^{-12}) = -27.631\ldots$, so that $\exp(\omega_\ell^{\text{baseline}}) = 10^{-12}$. Meanwhile, we set

$$\begin{aligned} \omega_{q,\ell} = \log \Bigg(&\sum_{j \in \mathcal{C}_q} \mathbb{1}\{Y_j \geq t_\ell\} \\ &- \sum_{j \in \mathcal{C}_q} \mathbb{1}\{Y_j \geq t_{\ell+1}\} \\ &- \sum_{\delta=1}^{m} \sum_{j \in \mathcal{C}_q} \mathbb{1}\{\Delta_j = \delta, Y_j = t_\ell\} \Bigg). \end{aligned}$$

To summarize, we have defined $d_{\delta,\ell}^{\text{cluster}}(q)$ and $c_\ell^{\text{cluster}}(q)$ as very simple neural nets with parameters γ and ω, respectively. From these we can back out the cluster-specific at risk count function

$$n_\ell^{\text{cluster}}(q) = \sum_{\delta=1}^{m} d_{\delta,\ell}^{\text{cluster}}(q) + c_\ell^{\text{cluster}}(q) + n_{\ell+1}^{\text{cluster}}(q).$$

Note that $n_\ell^{\text{cluster}}(q)$ depends on both γ and ω.

At this point, we treat the kernel function K as fixed, meaning that the neural net it depends on (f with parameters stored in $\widehat{\theta}$) is fixed. We also treat the cluster assignment as fixed. Then for any data point with feature vector x, its predicted event counts and at-risk counts are given by equations (20) and (21), which now are in terms of the neural net functions we just described with parameters γ and ω; we reproduce these below for the reader's convenience, where we now emphasize the dependence on γ and ω:

$$\begin{aligned} d_{\delta,\ell}^{\text{DKAJ}}(x;\gamma) &\triangleq \sum_{q \in \mathcal{Q}(x;\tau)} K(x, X_q) d_{\delta,\ell}^{\text{cluster}}(q;\gamma) \\ n_\ell^{\text{DKAJ}}(q;\gamma,\omega) &\triangleq \sum_{q \in \mathcal{Q}(x;\tau)} K(x, X_q) n_\ell^{\text{cluster}}(q;\gamma,\omega) \end{aligned}$$

Combining these two equations with equations (11) and (10), followed by plugging the resulting event-specific hazard function into the log of the likelihood function in equation (3), we get

$$\begin{aligned} \log \mathcal{L} = \sum_{\delta=1}^{m} \sum_{i=1}^{n} \Bigg[&\mathbb{1}\{\Delta_i = \delta\} \log \frac{d_{\delta,\kappa(Y_i)}^{\text{DKAJ}}(X_i;\gamma)}{n_\ell^{\text{DKAJ}}(X_i;\gamma,\omega)} \\ &- \sum_{\ell=1}^{\kappa(Y_i)} \frac{d_{\delta,\ell}^{\text{DKAJ}}(X_i;\gamma)}{n_\ell^{\text{DKAJ}}(X_i;\gamma,\omega)} \Bigg] + \text{constant}, \end{aligned} \tag{23}$$

which we can maximize using standard minibatch gradient descent (in which we would instead minimize the negative log likelihood loss $\mathbb{L}_{\text{NLL}} \triangleq -\frac{1}{n} \log \mathcal{L}$). Note that we do not need the leave-one-out strategy described in Section 3.1 because the functions $d_{\delta,\ell}^{\text{DKAJ}}(x;\gamma)$ and $n_\ell^{\text{DKAJ}}(x;\gamma,\omega)$ are not parameterized in terms of summations over the n training points (which had previously been the case for equations (12) and (13)). Adding a ranking loss (as to obtain the full DeepHit loss) is also possible; the idea is the same as what is presented in Appendix C.

Appendix F. Experimental Details

F.1. Datasets

In this section, we provide details of the datasets used in our experiments, including cohort selection criteria, feature definitions, and preprocessing. For all datasets, the preprocessing pipeline (encoding, scaling, imputation) is fit only on the training set and then applied consistently to the test set, to avoid data leakage. When applicable, for missing values on categorical and binary features, we use the mode of the corresponding features in the training for imputation; whereas average feature values in the training set is used for continuous variable imputation. Continuous variables are standardized, while categorical features are one-hot encoded.

PBC The PBC dataset is derived from a clinical trial of 1,945 patients with primary biliary cirrhosis (PBC). The full feature list we use in our modeling includes:

- `D-penicil`: indicator for treatment with D-penicillamine.
- `female`: binary sex indicator.
- `ascites`: presence of ascites.
- `hepatomegaly`: presence of hepatomegaly.
- `spiders`: presence of spider angiomas.
- `edema`: categorical indicator of edema (none, present without diuretics, present despite diuretics).
- `histologic`: histologic stage of disease.

- `serBilir`: serum bilirubin.
- `serChol`: serum cholesterol.
- `albumin`: serum albumin.
- `alkaline`: alkaline phosphatase.
- `SGOT`: serum glutamic-oxaloacetic transaminase.
- `platelets`: platelet count.
- `prothrombin`: prothrombin time.
- `age`: patient age at study entry.

Framingham The Framingham dataset originates from the Framingham Heart Study and includes 4,434 participants. Although longitudinal data are available, we restrict to the earliest observed features for each patient. We excluded patients with missing event time or censoring indicator. The full feature list we use in our modeling includes:

- `SEX`: Binary indicator of sex (0 = male, 1 = female).
- `CURSMOKE`: Current smoking status (1 = current smoker, 0 = non-smoker).
- `DIABETES`: Indicator of physician-diagnosed diabetes mellitus.
- `BPMEDS`: Indicator for use of antihypertensive medications at baseline.
- `PREVCHD`: History of coronary heart disease (CHD) prior to baseline.
- `PREVAP`: History of angina pectoris prior to baseline.
- `PREVMI`: History of myocardial infarction prior to baseline.
- `PREVSTRK`: History of stroke prior to baseline.
- `PREVHYP`: History of physician-diagnosed hypertension.
- `educ`: Educational attainment, coded as a categorical variable with four levels from 1 to 4.
- `TOTCHOL`: Total serum cholesterol (mg/dL).
- `AGE`: Age in years at baseline.
- `SYSBP`: Systolic blood pressure (mmHg).
- `DIABP`: Diastolic blood pressure (mmHg).
- `CIGPDAY`: Number of cigarettes smoked per day (self-reported).
- `BMI`: Body mass index (kg/m^2).
- `HEARTRTE`: Resting heart rate (beats per minute).
- `GLUCOSE`: Fasting blood glucose level (mg/dL).

SEER The SEER dataset is obtained from the U.S. Surveillance, Epidemiology, and End Results Program. While SEER spans multiple decades and includes diverse cancer types and both sexes, in this work we restrict to female patients diagnosed with breast cancer in the year 2010. For patients with multiple records, only the first entry per patient is retained. Patients without survival time are excluded. The outcome is breast cancer–specific mortality, with cardiovascular disease (CVD) mortality treated as the competing risk. The full feature list we use in our modeling includes:

- `Race and origin recode`: patient race and ethnicity.
- `Laterality`: laterality of the primary breast tumor (e.g., left, right).
- `Diagnostic Confirmation`: method of diagnosis (e.g., histology, cytology).
- `Histology recode`: histologic tumor classification.
- `Chemotherapy recode`: indicator for chemotherapy administration.
- `Radiation recode`: indicator for radiation therapy.
- `ER Status Recode Breast Cancer`: estrogen receptor status.
- `PR Status Recode Breast Cancer`: progesterone receptor status.
- `Sequence number`: sequence of the tumor relative to prior malignancies.
- `RX Summ--Surg Prim Site`: type of surgery performed on the primary tumor.
- `CS extension`: clinical extent of primary tumor spread.
- `CS lymph nodes`: involvement of regional lymph nodes.
- `CS mets at dx`: presence of distant metastasis at diagnosis.
- `Origin recode NHIA`: Hispanic origin classification.
- `Grade Recode`: histologic grade (I–IV or unknown).
- `Age recode with <1 year olds`: patient age grouped into categories (e.g., 01–04, 05–09, . . . , 85+).
- `Year of diagnosis`: restricted to 2010 in this study.
- `Total number of in situ/malignant tumors for patient`.
- `Total number of benign/borderline tumors for patient`.
- `CS tumor size`: tumor size in millimeters.
- `Regional nodes examined`: number of lymph nodes examined.
- `Regional nodes positive`: number of positive lymph nodes.

Synthetic We use the publicly available synthetic competing risks dataset by Lee et al. (2018).[10]

F.2. Evaluation Metrics

As we will soon see, both the integrated Brier score and time-dependent concordance index rely on a time grid. To ensure consistent comparison across models, evaluation is performed on a common time grid within a dataset across different data splits and hyperparameter configurations. For the evaluation time grid, we use 100 evenly spaced quantiles of the observed event times (see Kvamme and Borgan (2021), Section 3.1), starting from the minimum observed event times and truncated at the 90th percentile to reduce instability at the tail. This grid is constructed from the entire dataset for each experiment, instead of just from the training set, to make sure model comparison across different data splits is consistent. For discrete-time competing risks models (e.g., DeepHit and DKAJ), linear interpolation of the predicted CIFs is applied in order to evaluate at arbitrary time points.

Competing risks Brier score (BS) (Gerds and Kattan (2021), Section 5.4.2) Let $\delta \in [m]$ index the competing risks and $t \geq 0$. Given a cumulative incidence function (CIF) estimator $\widehat{F}_\delta(\cdot|x)$, the Brier score for event δ at time t is defined as

$$\begin{aligned}\mathrm{BS}_\delta(t) :=& \frac{1}{n}\sum_{i=1}^{n}\Bigg[\frac{(1-\widehat{F}_\delta(t|X_i))^2\mathbb{1}\{\Delta_i=\delta\}\mathbb{1}\{Y_i\leq t\}}{\widehat{S}_{\text{censor}}(Y_i)} \\ &+\frac{(\widehat{F}_\delta(t|X_i))^2\mathbb{1}\{\Delta_i\neq\delta,\,\Delta_i\neq 0\}\mathbb{1}\{Y_i\leq t\}}{\widehat{S}_{\text{censor}}(Y_i)} \\ &+\frac{(\widehat{F}_\delta(t|X_i))^2\mathbb{1}\{Y_i> t\}}{\widehat{S}_{\text{censor}}(t)}\Bigg],\end{aligned}$$

where $\widehat{S}_{\text{censor}}(\cdot)$ is the Kaplan-Meier estimate of the censoring distribution. Lower values indicate better performance. Notice that the competing risks Brier score is slightly different from the Brier score under the classic single-event survival analysis.

Competing risks integrated Brier score (IBS) To summarize performance across time, the competing risks Brier score is integrated over the evaluation time grid, producing the integrated Brier score (IBS). Lower values of IBS indicate better calibration and discrimination.

Time-dependent concordance index (C^{td}) We also report the time-dependent concordance index (Antolini et al., 2005), which measures a model's discriminative ability by assessing whether individuals with shorter survival times are assigned higher risk scores. When we are evaluating a model's performance for some event of interest using the C^{td}, we treat the occurrence of other events as censoring (i.e., the event indicator becomes censoring, and the event time of other events becomes the censoring time), which is a common strategy adopted in the literature. Higher values of C^{td} indicate better discriminative performance.

In summary, IBS (lower is better) emphasizes both calibration and discrimination, whereas C^{td} (higher is better) focuses only on discrimination. Both metrics are evaluated on the same truncated time grid described above.

F.3. Hyperparameter Grids and Optimization Details

We perform hyperparameter selection via grid search, with model selection based on the best validation performance (either average C^{td} or IBS across all events; see Appendix F.2).

Optimization and early stopping For deep models (DeepHit, DSM, NeuralFG, DKAJ) we use a maximum of 1000 epochs with early stopping (patience 10). DSM and NeuralFG (code from `NeuralFineGray`[11]) stop on the validation objective. DeepHit and DKAJ stop on the validation metric (IBS or C^{td}) aligned with the selection criterion. Appendix G.3 reports results when DeepHit and DKAJ also stop on validation objective; conclusions are unchanged. SurvivalBoost does not support early stopping in the reference implementation at the time of running experiments in this paper. Batch size is 1024 for all neural models (not applicable to Fine-Gray, cs-Cox, RSF-CR, and SurvivalBoost).

For each method, we provide additional details for replicability as follows:

Fine and Gray We use the implementation provided in the `cmprsk`[12] package written in R, which is called in Python via the `rpy2` package. There are no tunable hyperparameters.

10. https://github.com/chl8856/DeepHit
11. https://github.com/Jeanselme/NeuralFineGray/
12. https://cran.r-project.org/web/packages/cmprsk

Cause-specific Cox PH (cs-Cox) We use the `CSC` module implementation provided in the `riskRegression`[13] package written in R, which is called in Python via the `rpy2` package. Specifically, glmnet with 5-fold cross-validation on partial log-likelihood loss is applied to select the penalty.

Random survival forest for competing risks (RSF-CR) We use the `rfsrc` module implemented in the `randomForestSRC`[14] package written in R, which is called in Python via the `rpy2` package.

- Number of trees: $\{500, 750, 1000\}$
- Number of variables to possibly split at each node: $\{\text{None}, 8, 16\}$
- Minimum size of terminal node: $\{15, 30\}$
- Number of grid of time points over the observed event times: $\{0, 64, 128\}$, where 0 means using all observed event times

For the hyperparameters not mentioned above, default values provided by the package are applied.

DeepHit

- Number of layers: $\{2, 4\}$
- Hidden units per layer: $\{64, 128\}$
- Learning rate: $\{0.01, 0.001\}$
- α: $\{0, 0.001, 0.01\}$
- σ: $\{0.1, 1\}$
- Number of time steps L: $\{0, 64, 128\}$

The number of time steps being 0 means we use all possible observed event times in the training set, but an upper limit of 512 is used to prevent the neural net architecture from being too complex. When discretizing time into k steps, we use evenly spaced quantiles of the observed event times (see Kvamme and Borgan (2021), Section 3.1). This can lead to fewer unique time bins if many event times coincide.

Deep Survival Machines (DSM)

- Number of layers: $\{2, 4\}$
- Hidden units per layer: $\{64, 128\}$
- Learning rate: $\{0.01, 0.001\}$
- Number of mixture components k: $\{4, 8\}$
- Survival distribution family: {Weibull, LogNormal}
- Discount factor: $\{0.5, 1\}$

13. https://cran.r-project.org/web/packages/riskRegression
14. www.randomforestsrc.org

Neural Fine-Gray (NeuralFG)

- Number of encoder layers: $\{2, 4\}$
- Hidden units per encoder layer: $\{64, 128\}$
- Number of survival-specific layers: $\{1\}$
- Hidden units per survival-specific layer: $\{64, 128\}$
- Learning rate: $\{0.01, 0.001\}$
- Dropout: $\{0, 0.25, 0.5, 0.75\}$
- Activation: tanh

SurvivalBoost

- Learning rate: $\{0.01, 0.05, 0.1, 0.5\}$
- Number of boosting iterations: $\{20, 100, 200\}$
- Maximum tree depth: $\{-1, 4, 8, 16\}$
- Number of time grid steps: $\{64, 128\}$
- IPCW strategy: {alternating, Kaplan–Meier}

DKAJ

- Number of layers: $\{2, 4\}$
- Hidden units per layer: $\{64, 128\}$
- Learning rate: $\{0.01, 0.001\}$
- α: $\{0, 0.001, 0.01\}$
- σ: $\{0.1, 1\}$
- Number of time steps L: $\{0, 64, 128\}$
- γ: $\{0\}$
- β: $\{0.25, 0.5\}$
- Squared radius: $\{0.1\}$
- Minimum kernel weight (transformation of τ): $\{10^{-2}\}$

The number of time steps being 0 means we use all possible observed event times in the training set, but an upper limit of 512 is used to prevent the neural net architecture from being too complex. Moreover, we sample 10 random evaluation times per data point when using the TUNA neural net warm-start with SurvivalBoost (which leaf a data point is in per tree for SurvivalBoost depends on the evaluation time; we randomly choose different evaluation times).

Following Chen (2024b), τ is parameterized as a transformation of the minimum kernel weight, the smallest contribution $K(x, q)$ can make for any exemplar q. Specifically, $\tau = \sqrt{-\log(\text{min kernel weight})}$. In all experiments, we set the minimum kernel weight to 10^{-2}, consistent with prior work in survival kernels (Chen, 2024b), which provides a reasonable trade-off between smoothness and locality across datasets.

We use the TUNA warm start strategy to initialize DKAJ. In doing this warm start, we (i) sample 10 random evaluation times per individual when pairing with SurvivalBoost (its leaf assignment depends on the evaluation time), and (ii) perform a small inner sweep over architectural and cluster parameters

only for initialization, e.g., number of layers and hidden units, squared radius, and batch size. The best warm-start setting under the validation criterion is then frozen, and the main DKAJ likelihood training proceeds with the outer grid listed above. This two-stage scheme reduces the effective search space during the main optimization.

DKAJ Summary Function Fine-tuning (SFT)

- Fine-tuning learning rate: $\{0.01, 0.001, 0.0001\}$

Note that we only use the model with SFT if it results in improved average validation performance across different events; otherwise, we backtrack to the original DKAJ model before SFT.

Appendix G. Additional Experimental Results

G.1. Detailed Results Under Different Model Selection Criteria

Tables G.1 and G.2 report the full test performance of all models under two different model selection criteria: the best validation IBS and the best validation C^{td}, both of which are averaged across events, respectively. As discussed in footnote 9 in Section 4, the choice of validation criterion has a substantial impact on the observed test performance. When models are tuned to minimize average IBS across events, the resulting C^{td} scores are typically lower (Table G.1); conversely, tuning to maximize average C^{td} across events often yields inferior IBS values (Table G.2). This trade-off is consistent across datasets and event types, illustrating the tension between an evaluation metric that accounts for calibration (IBS) vs one that focuses only on discrimination (C^{td}) that is well documented in survival analysis.

We also investigate the effect of applying the optional summary function fine-tuning (SFT; Section 3.2) after training the DKAJ models. The corresponding results are reported in Tables G.1 and G.2. Overall, we observe that SFT often leads to further performance improvements for DKAJ across most datasets and event types. However, the magnitude of the improvement can be modest in some cases, and in rare instances, SFT slightly degrades performance on the test set.

G.2. Pairwise Performance Significance

To assess the statistical significance of performance differences among models, we conduct paired, two-sided Wilcoxon signed-rank tests for both metrics: IBS and C^{td}. Each test pools 80 paired observations (4 datasets $\times$ 10 random splits $\times$ 2 events), with pairs matched on dataset, split, and event to ensure comparable conditions across methods.

To align evaluation with the selection objective, we run two independent analyses: (i) for the IBS tests, model instances are chosen based on the best validation IBS; (ii) for the Ctd tests, model instances are chosen based on the best validation Ctd. As discussed in the main text, the validation criterion can influence relative rankings, so separating the analyses avoids conflating selection effects across metrics.

For each of the evaluation metrics (IBS or C^{td}), we conduct the test separately using the following steps:

1. For each unordered method pair and for the relevant metric/selection setting, we compute paired differences across the 80 matched observations (ignoring ties).
2. We apply the Wilcoxon signed-rank test (two-sided) to obtain one p-value per unordered pair of methods.
3. To control the family-wise error rate within each metric, we apply the Holm–Bonferroni correction across all unordered pairs, yielding adjusted p-values.

The complete pairwise outcomes are provided in Tables G.3 (IBS, models selected by IBS) and G.4 (C^{td}, models selected by C^{td}). Across metrics, DKAJ is broadly competitive with the strongest baselines and exhibits statistically significant improvements over several classical and deep models in numerous pairings. Differences among top-performing neural and ensemble methods (e.g., SurvivalBoost and NeuralFG) are often not statistically significant after Holm correction, indicating comparable performance levels within this group. These results reinforce the aggregate findings presented in the main paper and highlight the stability of DKAJ's performance across both discrimination and calibration metrics under consistent statistical testing.

G.3. Performance Using Alternative Early Stopping

As described in Appendix F.3, the deep learning–based methods differ slightly in their early stopping criteria. Specifically, DeepHit and DKAJ employ *metric-aligned early stopping*, where training stops once the validation performance metric (IBS or C^{td}) plateaus for 10 epochs. In contrast, DSM

Table G.1: Test set IBS and C^{td} (mean ± std. dev. across 10 random splits) for primary and competing events (best score in **bold**, 2nd best in blue). Model selection done by the best validation IBS; the IBS columns are shaded to highlight the results.

Dataset	Method	IBS (↓)		C^{td} (↑)	
		Primary	Competing	Primary	Competing
PBC	Fine-Gray	0.1086±0.0047	0.0401±0.0046	0.8212±0.0119	0.8769±0.0203
	cs-Cox	0.1052±0.0051	0.0391±0.0047	0.8295±0.0102	0.9094±0.0132
	RSF-CR	0.0989±0.0035	0.0412±0.0049	0.8584±0.0082	0.8803±0.0143
	DeepHit	0.1259±0.0121	0.0552±0.0074	0.8243±0.0275	0.8050±0.0765
	DSM	0.1103±0.0055	0.0419±0.0041	0.8331±0.0103	0.8843±0.0204
	NeuralFG	0.1055±0.0061	0.0395±0.0056	0.8350±0.0151	0.9060±0.0133
	SurvivalBoost	**0.0898±0.0046**	**0.0322±0.0041**	**0.8745±0.0071**	**0.9419±0.0072**
	DKAJ	0.1031±0.0071	0.0394±0.0057	0.8242±0.0215	0.7885±0.0825
	DKAJ + SFT	0.1037±0.0072	0.0384±0.0058	0.8351±0.0177	0.8792±0.0444
Framingham	Fine-Gray	0.0819±0.0037	**0.0566±0.0035**	0.7733±0.0106	0.7144±0.0182
	cs-Cox	**0.0814±0.0039**	**0.0566±0.0034**	**0.7753±0.0105**	**0.7160±0.0173**
	RSF-CR	0.0815±0.0031	0.0573±0.0036	0.7691±0.0097	0.6924±0.0122
	DeepHit	0.0941±0.0049	0.0645±0.0055	0.7099±0.0146	0.6464±0.0224
	DSM	0.0847±0.0045	0.0610±0.0046	0.7657±0.0154	0.7119±0.0183
	NeuralFG	0.0880±0.0035	0.0624±0.0039	0.7060±0.0706	0.5164±0.1516
	SurvivalBoost	0.0824±0.0038	0.0573±0.0036	0.7640±0.0136	0.6975±0.0158
	DKAJ	0.0844±0.0036	0.0578±0.0034	0.7613±0.0155	0.6870±0.0192
	DKAJ + SFT	0.0853±0.0031	0.0579±0.0035	0.7629±0.0158	0.6882±0.0187
SEER	Fine-Gray	0.0637±0.0013	0.0165±0.0010	0.8151±0.0021	0.8478±0.0099
	cs-Cox	0.0638±0.0013	**0.0164±0.0009**	0.7630±0.0085	0.8415±0.0132
	RSF-CR	0.0599±0.0013	0.0167±0.0010	0.8346±0.0033	0.8244±0.0139
	DeepHit	0.0721±0.0012	0.0228±0.0017	0.8043±0.0140	0.8134±0.0494
	DSM	0.0613±0.0011	0.0171±0.0008	0.7809±0.0096	0.8407±0.0089
	NeuralFG	0.0608±0.0014	0.0166±0.0009	0.7736±0.0094	0.8360±0.0154
	SurvivalBoost	**0.0584±0.0014**	0.0165±0.0008	**0.8424±0.0029**	**0.8558±0.0100**
	DKAJ	0.0609±0.0011	0.0167±0.0011	0.8277±0.0043	0.8250±0.0153
	DKAJ + SFT	0.0792±0.0138	0.0197±0.0041	0.8274±0.0038	0.8316±0.0182
Synthetic	Fine-Gray	0.1851±0.0035	0.1827±0.0027	0.5823±0.0051	0.5917±0.0071
	cs-Cox	0.1851±0.0035	0.1827±0.0027	0.5809±0.0051	0.5902±0.0073
	RSF-CR	0.1745±0.0033	0.1739±0.0027	0.7217±0.0076	0.7169±0.0040
	DeepHit	0.1732±0.0046	0.1702±0.0026	0.7214±0.0181	0.7233±0.0127
	DSM	0.1743±0.0034	0.1715±0.0028	0.7255±0.0067	0.7295±0.0044
	NeuralFG	**0.1631±0.0033**	**0.1606±0.0027**	**0.7479±0.0069**	**0.7518±0.0041**
	SurvivalBoost	0.1734±0.0034	0.1712±0.0025	0.7062±0.0083	0.7120±0.0034
	DKAJ	0.1672±0.0039	0.1647±0.0029	0.7318±0.0083	0.7346±0.0052
	DKAJ + SFT	0.1777±0.0086	0.1729±0.0056	0.7272±0.0086	0.7310±0.0060

Table G.2: Test set IBS and C^{td} (mean ± std. dev. across 10 random splits) for primary and competing events (best score in **bold**, 2nd best in blue). Model selection done by the best validation C^{td}; the C^{td} columns are shaded to highlight the results.

Dataset	Method	IBS (↓)		C^{td} (↑)	
		Primary	Competing	Primary	Competing
PBC	Fine-Gray	0.1086±0.0047	0.0401±0.0046	0.8212±0.0119	0.8769±0.0203
	cs-Cox	0.1052±0.0051	0.0391±0.0047	0.8295±0.0102	0.9094±0.0132
	RSF-CR	0.0995±0.0035	0.0414±0.0048	0.8584±0.0089	0.8746±0.0163
	DeepHit	0.1308±0.0052	0.0785±0.0086	0.8407±0.0131	0.9060±0.0130
	DSM	0.1185±0.0099	0.0404±0.0049	0.8319±0.0115	0.9039±0.0189
	NeuralFG	0.1060±0.0058	0.0390±0.0046	0.8363±0.0150	0.9120±0.0149
	SurvivalBoost	**0.0924±0.0085**	**0.0329±0.0050**	**0.8719±0.0107**	**0.9360±0.0114**
	DKAJ	0.1248±0.0084	0.0415±0.0055	0.8413±0.0105	0.9039±0.0170
	DKAJ + SFT	0.1241±0.0075	0.0409±0.0051	0.8415±0.0102	0.9037±0.0181
Framingham	Fine-Gray	0.0819±0.0037	**0.0566±0.0035**	0.7733±0.0106	0.7144±0.0182
	cs-Cox	**0.0814±0.0039**	**0.0566±0.0034**	**0.7753±0.0105**	**0.7160±0.0173**
	RSF-CR	0.0818±0.0032	0.0574±0.0036	0.7720±0.0098	0.6982±0.0119
	DeepHit	0.1168±0.0034	0.0880±0.0018	0.7423±0.0198	0.6957±0.0195
	DSM	0.0890±0.0038	0.0689±0.0053	0.7664±0.0170	0.7095±0.0184
	NeuralFG	0.0992±0.0140	0.0628±0.0037	0.6833±0.0619	0.6978±0.0287
	SurvivalBoost	0.0830±0.0046	0.0571±0.0036	0.7645±0.0152	0.7031±0.0156
	DKAJ	0.0899±0.0047	0.0579±0.0038	0.7677±0.0138	0.7076±0.0209
	DKAJ + SFT	0.0928±0.0108	0.0588±0.0057	0.7681±0.0151	0.7088±0.0192
SEER	Fine-Gray	0.0637±0.0013	0.0165±0.0010	0.8151±0.0021	0.8478±0.0099
	cs-Cox	0.0638±0.0013	**0.0164±0.0009**	0.7630±0.0085	0.8415±0.0132
	RSF-CR	0.0607±0.0016	0.0168±0.0010	0.8350±0.0030	0.8301±0.0137
	DeepHit	0.0914±0.0077	0.0550±0.0175	0.8239±0.0028	0.8548±0.0120
	DSM	0.0626±0.0014	0.0186±0.0019	0.7807±0.0088	0.8422±0.0119
	NeuralFG	0.0611±0.0016	0.0166±0.0009	0.7743±0.0090	0.8362±0.0138
	SurvivalBoost	**0.0587±0.0014**	**0.0164±0.0009**	**0.8418±0.0038**	0.8583±0.0081
	DKAJ	0.0691±0.0048	0.0169±0.0010	0.8333±0.0034	**0.8598±0.0072**
	DKAJ + SFT	0.0868±0.0181	0.0216±0.0045	0.8371±0.0043	0.8597±0.0070
Synthetic	Fine-Gray	0.1851±0.0035	0.1827±0.0027	0.5823±0.0051	0.5917±0.0071
	cs-Cox	0.1851±0.0035	0.1827±0.0027	0.5809±0.0051	0.5902±0.0073
	RSF-CR	0.1752±0.0035	0.1742±0.0027	0.7224±0.0071	0.7203±0.0043
	DeepHit	0.2307±0.0071	0.2265±0.0088	0.7401±0.0067	0.7437±0.0043
	DSM	0.1750±0.0036	0.1725±0.0035	0.7280±0.0055	0.7320±0.0042
	NeuralFG	**0.1634±0.0039**	**0.1612±0.0031**	**0.7481±0.0073**	**0.7513±0.0045**
	SurvivalBoost	0.1753±0.0033	0.1736±0.0026	0.7160±0.0083	0.7190±0.0039
	DKAJ	0.1729±0.0048	0.1700±0.0040	0.7384±0.0067	0.7435±0.0041
	DKAJ + SFT	0.1783±0.0033	0.1757±0.0031	0.7388±0.0072	0.7439±0.0044

Table G.3: Paired, two-sided Wilcoxon signed-rank tests on **IBS** (lower is better), pooled over 4 datasets × 10 splits × 2 events (80 pairs per comparison; pairs matched on dataset, split, and event). Cells show the row–vs–column direction and Holm–Bonferroni–corrected significance across all unordered method pairs for IBS: ↓ = row lower (better), ↑ = row higher (worse); ***: $\tilde{p} < 0.001$, **: $\tilde{p} < 0.01$, *: $\tilde{p} < 0.05$, ≈: not significant, where $\tilde{p}$ denotes the Holm-adjusted p-value.

	Fine-Gray	*cs-Cox*	*RSF-CR*	*DeepHit*	*DSM*	*NeuralFG*	*SurvivalBoost*	*DKAJ*	*DKAJ+SFT*
Fine-Gray	-	$\uparrow^{**}$	$\uparrow^{***}$	≈	≈	≈	$\uparrow^{***}$	$\uparrow^{**}$	≈
cs-Cox	$\downarrow^{**}$	—	$\uparrow^{***}$	$\downarrow^{*}$	≈	≈	$\uparrow^{***}$	$\uparrow^{*}$	≈
RSF-CR	$\downarrow^{***}$	$\downarrow^{***}$	—	$\downarrow^{***}$	$\downarrow^{***}$	≈	$\uparrow^{***}$	≈	$\downarrow^{**}$
DeepHit	≈	$\uparrow^{*}$	$\uparrow^{***}$	—	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{**}$
DSM	≈	≈	$\uparrow^{***}$	$\downarrow^{***}$	—	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	≈
NeuralFG	≈	≈	≈	$\downarrow^{***}$	$\downarrow^{***}$	—	≈	≈	$\downarrow^{*}$
SurvivalBoost	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$	≈	—	≈	$\downarrow^{***}$
DKAJ	$\downarrow^{**}$	$\downarrow^{*}$	≈	$\downarrow^{***}$	$\downarrow^{***}$	≈	≈	—	$\downarrow^{***}$
DKAJ+SFT	≈	≈	$\uparrow^{**}$	$\downarrow^{**}$	≈	$\uparrow^{*}$	$\uparrow^{***}$	$\uparrow^{***}$	—

Table G.4: Paired, two-sided Wilcoxon signed-rank tests on $C^{\mathbf{td}}$ (higher is better), pooled over 4 datasets × 10 splits × 2 events (80 pairs per comparison; pairs matched on dataset, split, and event). Cells show the row–vs–column direction and Holm–Bonferroni–corrected significance across all unordered method pairs for C^{td}: ↑ = row higher (better), ↓ = row lower (worse); ***: $\tilde{p} < 0.001$, **: $\tilde{p} < 0.01$, *: $\tilde{p} < 0.05$, ≈: not significant, where $\tilde{p}$ denotes the Holm-adjusted p-value.

	Fine-Gray	*cs-Cox*	*RSF-CR*	*DeepHit*	*DSM*	*NeuralFG*	*SurvivalBoost*	*DKAJ*	*DKAJ+SFT*
Fine-Gray	—	≈	$\downarrow^{**}$	$\downarrow^{**}$	≈	≈	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$
cs-Cox	≈	—	$\downarrow^{**}$	$\downarrow^{**}$	$\downarrow^{**}$	≈	$\downarrow^{***}$	$\downarrow^{***}$	$\downarrow^{***}$
RSF-CR	$\uparrow^{**}$	$\uparrow^{**}$	—	≈	≈	≈	$\downarrow^{**}$	$\downarrow^{***}$	$\downarrow^{***}$
DeepHit	$\uparrow^{**}$	$\uparrow^{**}$	≈	—	≈	≈	≈	$\downarrow^{*}$	$\downarrow^{**}$
DSM	≈	$\uparrow^{**}$	≈	≈	—	≈	$\downarrow^{**}$	$\downarrow^{***}$	$\downarrow^{***}$
NeuralFG	≈	≈	≈	≈	≈	—	$\downarrow^{**}$	$\downarrow^{**}$	$\downarrow^{**}$
SurvivalBoost	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{**}$	≈	$\uparrow^{**}$	$\uparrow^{**}$	—	≈	≈
DKAJ	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{*}$	$\uparrow^{***}$	$\uparrow^{**}$	≈	—	≈
DKAJ+SFT	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{***}$	$\uparrow^{**}$	$\uparrow^{***}$	$\uparrow^{**}$	≈	≈	—

and NeuralFG follow *objective-based early stopping*, terminating training when the learning objective on the validation set ceases to improve for 10 epochs. Although both approaches are standard, this introduces a minor inconsistency across models in how the stopping signal is defined.

To assess the sensitivity of our results to the stopping criterion, we retrained DeepHit and DKAJ using the same objective-based stopping rule as DSM and NeuralFG (validation objective, patience 10). For these runs, model checkpoints were saved according to the best validation objective, but the final model

selected for reporting test performance, among all hyperparameter configurations, remained based on the best validation metric (IBS or C^{td}), consistent with our main protocol. Tables G.5 and G.6 compare the two stopping strategies. Across datasets and events, the differences are numerically negligible and do not alter conclusions. We therefore retain metric-aligned early stopping for DeepHit and DKAJ in the main experiments.

G.4. Ablation Study

We ablate two components of DKAJ to assess their necessity: (i) the leave-one-out (LOO) likelihood in (17), replaced by the standard likelihood in (16) to form *noLOO*; and (ii) the TUNA warm start, replaced by PyTorch's default initialization to form *noTUNA*. For each ablation we toggle only the targeted component and keep the architecture, hyperparameter grids, early-stopping policy, and search protocol identical to DKAJ (see Appendix F.3 for details on grids and optimization).

To avoid mixing selection criteria across metrics, we use metric-aligned model selection: IBS values are reported from the model chosen by the best validation IBS, and C^{td} values are reported from the model chosen by the best validation C^{td}. We report results in terms of mean ± standard deviation over 10 random splits.

When training without the LOO formulation, validation performance typically deteriorates early, indicating overfitting—consistent with the intended regularizing role of excluding each subject's own label from its kernel-weighted hazards. The TUNA warm start, by contrast, primarily stabilizes optimization and improves the initial embedding geometry. As noted in Appendix F.3, this two-stage scheme also reduces the effective hyperparameter search space during the main training.

Table G.7 summarizes the ablations on three real-world datasets (PBC, Framingham, SEER) and one synthetic dataset. On the real-world datasets, removing either component generally worsens performance in most cases. In particular, noLOO shows lower C^{td} and higher IBS on SEER and Framingham, and exhibits increased variability on PBC, while noTUNA yields smaller changes relative to DKAJ. On the synthetic dataset, noLOO substantially degrades both metrics, whereas noTUNA is comparable to (and in some entries slightly better than) DKAJ.

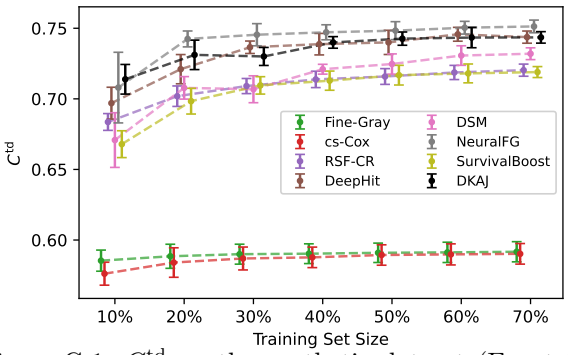

Figure G.1: C^{td} on the synthetic dataset (Event 2) as training set size varies; 30% held-out test.

Overall, these results indicate that both components contribute to the robustness of DKAJ.

G.5. Varying Training Size

To examine robustness under different data availability, we vary the proportion of data used for training from 10% to 70%, reserving the remaining 30% as a held-out test set. Each configuration includes both training and validation data within the specified proportion. Results are averaged over 10 random splits.

Figures 1 and G.1 show the C^{td} for the two competing events. Across methods, performance generally improves as the training size increases and stabilizes beyond approximately 40–50%. DKAJ scales smoothly with data size and follows similar performance trends to other neural-network-based models (DeepHit, NeuralFG) and the ensemble baseline (SurvivalBoost). We observe no indication of overfitting even in smaller training regimes, confirming that DKAJ maintains stable and data-efficient behavior across varying cohort sizes.

G.6. Computation Cost

We report wall–clock training time and model size on the Framingham dataset. Times are averaged over 10 random splits (mean ± std), and reflect the *total* time per split including the full hyperparameter sweep under IBS–based model selection. Model size is summarized as the number of learnable neural–network parameters; for DKAJ, we also report the number of entries in the (deterministic) summary tables used at inference (not trainable and not part of backprop).

Training time Table G.8 shows the total training time per method. DKAJ is the same order of mag-

Table G.5: Comparison of metric-aligned vs. objective-based early stopping when models are selected by best validation IBS. For DeepHit and DKAJ, "(ObjStop)" denotes runs stopped based on the validation objective (patience 10), as used for DSM and NeuralFG. Model selection for reporting test results remains based on best validation IBS; the IBS columns are shaded to highlight the results. Differences due to the stopping rule are negligible.

Dataset	**Method**	**IBS** (↓)		$C^{\mathbf{td}}$ (↑)	
		Primary	Competing	Primary	Competing
PBC	DeepHit	0.1259±0.0121	0.0552±0.0074	0.8243±0.0275	0.8050±0.0765
	DeepHit (ObjStop)	0.1337±0.0099	0.0652±0.0079	0.8280±0.0154	0.8035±0.0757
	DKAJ	0.1031±0.0071	0.0394±0.0057	0.8242±0.0215	0.7885±0.0825
	DKAJ (ObjStop)	0.1093±0.0080	0.0401±0.0056	0.8043±0.0466	0.7996±0.1260
Framingham	DeepHit	0.0941±0.0049	0.0645±0.0055	0.7099±0.0146	0.6464±0.0224
	DeepHit (ObjStop)	0.1106±0.0080	0.0669±0.0076	0.7006±0.0187	0.6549±0.0238
	DKAJ	0.0844±0.0036	0.0578±0.0034	0.7613±0.0155	0.6870±0.0192
	DKAJ (ObjStop)	0.0836±0.0031	0.0578±0.0034	0.7651±0.0148	0.6867±0.0326
SEER	DeepHit	0.0721±0.0012	0.0228±0.0017	0.8043±0.0140	0.8134±0.0494
	DeepHit (ObjStop)	0.0753±0.0022	0.0237±0.0020	0.8116±0.0179	0.8435±0.0125
	DKAJ	0.0609±0.0011	0.0167±0.0011	0.8277±0.0043	0.8250±0.0153
	DKAJ (ObjStop)	0.0612±0.0013	0.0167±0.0010	0.8297±0.0054	0.8354±0.0150
Synthetic	DeepHit	0.1732±0.0046	0.1702±0.0026	0.7214±0.0181	0.7233±0.0127
	DeepHit (ObjStop)	0.1733±0.0046	0.1695±0.0027	0.7334±0.0073	0.7376±0.0063
	DKAJ	0.1672±0.0039	0.1647±0.0029	0.7318±0.0083	0.7346±0.0052
	DKAJ (ObjStop)	0.1667±0.0037	0.1640±0.0026	0.7320±0.0068	0.7367±0.0056

Table G.6: Comparison of metric-aligned vs. objective-based early stopping when models are selected by best validation C^{td}. For DeepHit and DKAJ, "(ObjStop)" denotes runs stopped based on the validation objective (patience 10), as used for DSM and NeuralFG. Model selection for reporting test results remains based on best validation C^{td}; the C^{td} columns are shaded to highlight the results. Differences due to the stopping rule are negligible.

Dataset	**Method**	**IBS** (↓)		$C^{\mathbf{td}}$ (↑)	
		Primary	Competing	Primary	Competing
PBC	DeepHit	0.1308±0.0052	0.0785±0.0086	0.8407±0.0131	0.9060±0.0130
	DeepHit (ObjStop)	0.1343±0.0062	0.0892±0.0127	0.8322±0.0145	0.8946±0.0185
	DKAJ	0.1248±0.0084	0.0415±0.0055	0.8413±0.0105	0.9039±0.0170
	DKAJ (ObjStop)	0.1240±0.0073	0.0417±0.0058	0.8406±0.0077	0.9044±0.0140
Framingham	DeepHit	0.1168±0.0034	0.0880±0.0018	0.7423±0.0198	0.6957±0.0195
	DeepHit (ObjStop)	0.1153±0.0045	0.0848±0.0085	0.7423±0.0178	0.6907±0.0212
	DKAJ	0.0899±0.0047	0.0579±0.0038	0.7677±0.0138	0.7076±0.0209
	DKAJ (ObjStop)	0.0900±0.0048	0.0579±0.0037	0.7673±0.0119	0.7112±0.0192
SEER	DeepHit	0.0914±0.0077	0.0550±0.0175	0.8239±0.0028	0.8548±0.0120
	DeepHit (ObjStop)	0.0863±0.0089	0.0483±0.0162	0.8244±0.0042	0.8548±0.0095
	DKAJ	0.0691±0.0048	0.0169±0.0010	0.8333±0.0034	0.8598±0.0072
	DKAJ (ObjStop)	0.0692±0.0048	0.0169±0.0010	0.8334±0.0045	0.8566±0.0113
Synthetic	DeepHit	0.2307±0.0071	0.2265±0.0088	0.7401±0.0067	0.7437±0.0043
	DeepHit (ObjStop)	0.2299±0.0129	0.2239±0.0088	0.7408±0.0072	0.7439±0.0061
	DKAJ	0.1729±0.0048	0.1700±0.0040	0.7384±0.0067	0.7435±0.0041
	DKAJ (ObjStop)	0.1744±0.0037	0.1722±0.0027	0.7398±0.0067	0.7445±0.0044

Table G.7: Ablation results with metric-aligned reporting: IBS is selected by best validation IBS; C^{td} is selected by best validation C^{td}. Mean ± std. dev. over 10 splits.

Dataset	Method	IBS (↓)		C^{td} (↑)	
		Primary	Competing	Primary	Competing
PBC	DKAJ	0.1031±0.0071	0.0394±0.0057	**0.8413±0.0105**	**0.9039±0.0170**
	DKAJ noLOO	**0.1012±0.0077**	**0.0393±0.0056**	0.8382±0.0137	0.8827±0.0371
	DKAJ noTUNA	0.1049±0.0054	0.0399±0.0054	0.8399±0.0114	0.8977±0.0258
Framingham	DKAJ	**0.0844±0.0036**	**0.0578±0.0034**	**0.7677±0.0138**	**0.7076±0.0209**
	DKAJ noLOO	0.0869±0.0056	0.0579±0.0035	0.7644±0.0131	0.6973±0.0203
	DKAJ noTUNA	0.0848±0.0030	**0.0578±0.0037**	0.7644±0.0149	0.7041±0.0191
SEER	DKAJ	**0.0609±0.0011**	**0.0167±0.0011**	**0.8333±0.0034**	**0.8598±0.0072**
	DKAJ noLOO	0.0637±0.0019	0.0170±0.0009	0.8175±0.0068	0.8313±0.0129
	DKAJ noTUNA	0.0617±0.0013	0.0169±0.0010	0.8256±0.0040	0.8585±0.0076
Synthetic	DKAJ	0.1672±0.0039	0.1647±0.0029	0.7384±0.0067	0.7435±0.0041
	DKAJ noLOO	0.1808±0.0033	0.1790±0.0029	0.6174±0.0098	0.6279±0.0098
	DKAJ noTUNA	**0.1660±0.0037**	**0.1637±0.0028**	**0.7427±0.0071**	**0.7461±0.0048**

nitude as other deep baselines (faster than NeuralFG under our grid), and slower than shallow or classical models. Table G.9 decomposes DKAJ's time into: SurvivalBoost pretraining (used by the warm start), the TUNA warm–up network fitting, and the main DKAJ training. The warm–up overhead is small; the total is dominated by SurvivalBoost and the main DKAJ phase.

Table G.8: Average total training time per split (minutes; mean ± std over 10 splits) on Framingham. Times include the full hyperparameter sweep.

Method	Time (minutes)
Fine-Gray	0.064 ± 0.015
cs-Cox	0.006 ± 0.000
RSF-CR	0.142 ± 0.072
DeepHit	1.279 ± 0.461
DSM	7.894 ± 1.397
NeuralFG	32.176 ± 29.290
SurvivalBoost	13.526 ± 1.333
DKAJ (total)	22.813 ± 2.433

Model size and storage Table G.10 reports the number of learnable parameters for each neural baseline and, for DKAJ, the number of entries in the summary tables used for cluster/time lookups at inference. These summary tables scale with the number of clusters and the time grid, but they are fixed (non–trainable) and do not contribute to backprop memory. The trainable parameter count of DKAJ is comparable to other deep baselines.

Table G.9: DKAJ training time decomposition on Framingham (minutes; mean ± std over 10 splits). "SurvivalBoost" is used by the warm start; "TUNA warm–up" initializes neural net weights to approximate SurvivalBoost prediction; "DKAJ (main) is the main training using a DeepHit loss function.

Method	Time (minutes)
SurvivalBoost	13.526 ± 1.333
DKAJ (TUNA warm-up)	0.295 ± 0.091
DKAJ (main)	8.992 ± 1.883
DKAJ (total)	22.813 ± 2.433

Appendix H. Additional Visualization that Help with Interpretation

H.1. Individual-Level Visualizations

As discussed in Sections 3.2 and 4, a trained DKAJ estimator inherently supports individual-level interpretation. We illustrate this using the Framingham dataset, although the same approach applies to any dataset.

For a given test data point, the DKAJ model produces predicted CIFs for each competing event, along with weights that quantify the contribution of differ-

Table G.10: Model size (learnable parameters) and, for DKAJ, the number of entries in the deterministic summary tables used at inference. Values are mean ± std across hyperparameter settings and splits.

Method	**No. parameters** (neural net)	**No. entries in DKAJ summary functions** (equation (18), (19))
DeepHit	7581k ± 25k	∼
DSM	1560k ± 0k	∼
NeuralFG	3216k ± 0k	∼
DKAJ	2547k ± 957k	71373k ± 10275k

ent clusters to the prediction. Figure H.1 displays the CIFs (i.e., Aalen–Johansen curves) of the 5 clusters with the highest weights for a randomly chosen test patient, together with the model's overall prediction for that individual. This allows us to identify which clusters are most influential in shaping the prediction.

To better understand the defining characteristics of these influential clusters, Figure H.2 shows a feature-level heatmap summarizing the distribution of features across the same 5 clusters. By examining these cluster profiles, we can link predictive behavior to specific clinical factors.

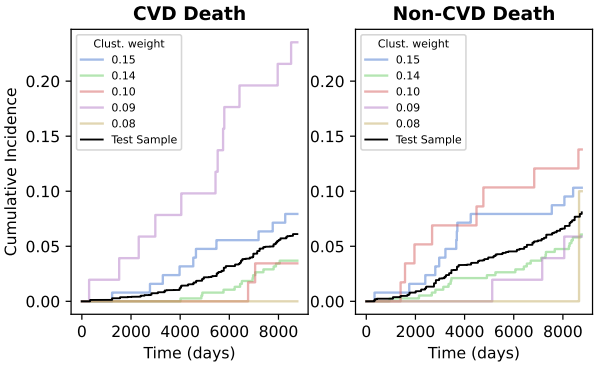

Figure H.1: (Framingham, individual-level) Top 5 clusters with the highest weights assigned to a randomly chosen test patient. We show the AJ curves for the 5 clusters as well as the predicted CIF for this test patient. Clusters correspond across the two plots in this figure as well as in Figure H.2.

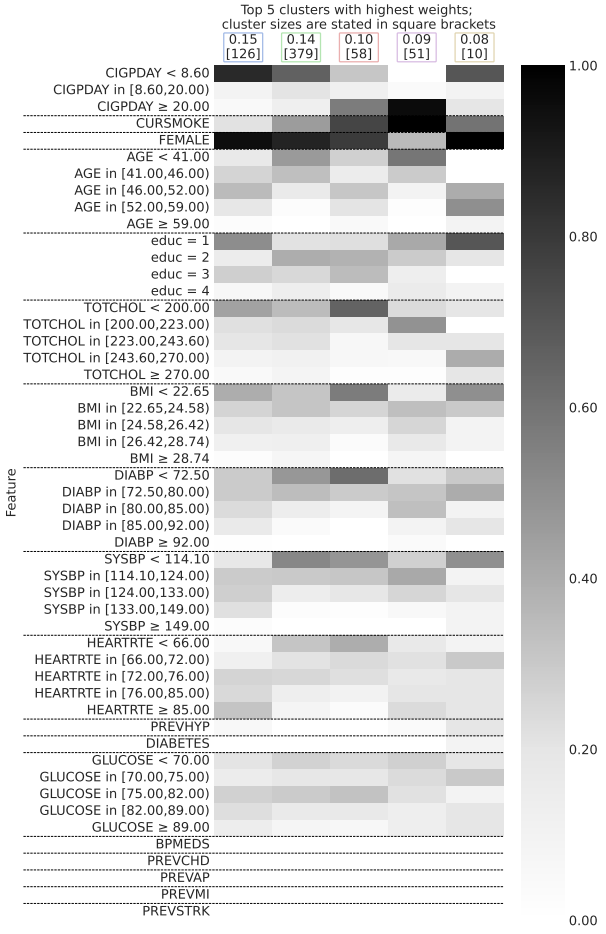

Figure H.2: (Framingham, individual-level) Feature heatmap summarizing distributions of variables in the same 5 clusters as in Figure H.1. Darker shades mean higher feature values or frequencies.

Probabilities of events happening earliest The probability of event $\delta \in [m]$ happening earliest for a test patient with feature vector x is given by

$$\begin{aligned}\mathbb{P}(\Delta^* = \delta \mid X = x) &= \mathbb{P}(\Delta^* = \delta, T \leq \infty \mid X = x) \\ &= F_\delta(\infty|x),\end{aligned}$$

i.e., the CIF for event δ evaluated at time ∞.

In practice, we do not actually estimate the CIFs arbitrarily accurately all the way to time ∞ and can instead just extrapolate from the maximum observed time in the training data ($t_{\max}$). For example, we can extrapolate using forward-filling (i.e., just approximate $F_\delta(\infty|x)$ with $F_\delta(t_{\max}|x)$). However, in doing so, when we sum the CIFs of the different events each evaluated at time $t_{\max}$, we are not guaranteed to get 1. We simply re-normalize to get the distribution to sum to 1 as to approximate the probability of each event occurring earliest. In other words, we approximate the probability of event δ happening earliest for feature vector x by

$$\mathbb{P}(\Delta^* = \delta \mid X = x) \approx \frac{F_\delta(t_{\max}|x)}{\sum_{\delta'=1}^{m} F_{\delta'}(t_{\max}|x)}.$$

For example, for the same random test patient that appears in Figures H.1 and H.2, at $t_{\max}$, the predicted CIFs for CVD death and non-CVD death are 0.0611 and 0.0808, respectively. The probability that CVD death occurs earliest is then estimated as $\frac{0.0611}{0.0611+0.0808} = 43.04\%$, while the probability of non-CVD death occurring first is $\frac{0.0808}{0.0611+0.0808} = 56.96\%$.

Median time until an event happens given that the event is the earliest to happen It is sometimes also of interest to compute the median time until a given event happens for a data point. However, in the competing risks setup when $m > 1$, for a particular data point, it could be that a specific event cannot happen (i.e., the data point's median time until the event is ∞). Instead, we can look at the median time until a given event δ happens, *conditioned on event δ being the earliest to happen for a test patient with feature vector x*. To estimate this, note that the CDF of the time until event δ happens, conditioned on δ being the earliest event to happen for feature vector x is

$$\begin{aligned}
&\mathbb{P}(T \le t \mid X = x, \Delta^* = \delta) \\
&\quad = \frac{\mathbb{P}(T \le t, \Delta^* = \delta \mid X = x)}{\mathbb{P}(\Delta^* = \delta \mid X = x)} \\
&\quad = \frac{\mathbb{P}(T \le t, \Delta^* = \delta \mid X = x)}{\mathbb{P}(T \le \infty, \Delta^* = \delta \mid X = x)} \\
&\quad = \frac{F_\delta(t|x)}{F_\delta(\infty|x)} \\
&\quad \approx \frac{F_\delta(t|x)}{F_\delta(t_{\max}|x)}.
\end{aligned}$$

The time t at which this CDF crosses 1/2 is a median time estimate. Returning to the same test patient as in Figures H.1 and H.2, by approximately finding t for which $\frac{F_\delta(t|x)}{F_\delta(t_{\max}|x)}$ crosses 1/2 for each event δ, we estimate the median time until CVD death assuming the CVD death is the earliest to happen as 2,888 days for the test subject. For non-CVD death, we instead get 2,562 days.

H.2. Additional Cluster-Level Visualizations

We provide supplemental cluster-level views to contextualize the groups discovered by DKAJ. Specifically, we (i) present cluster-wise CIFs and feature heatmaps for PBC and SEER, and (ii) visualize the similarity structure via a kernel-similarity heatmap for the Framingham dataset. Unless noted, clusters are ranked by size.

Cluster visualization for PBC and SEER In Section 4, we illustrated cluster-level visualizations using the Framingham dataset. We now provide similar analyses for the PBC and SEER datasets.

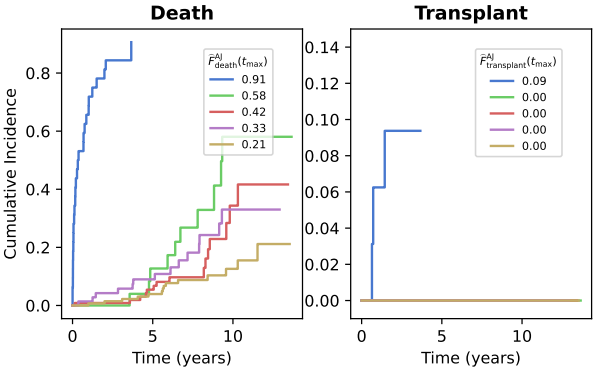

Figure H.3: (PBC) CIFs for the largest 5 clusters (we then sort these 5 clusters in decreasing order by the estimated probability of death happening within the maximum observed time). Clusters correspond across the two plots in this figure as well as in Figure H.4.

Figures H.3 and H.4 present the largest five clusters identified in the PBC dataset. The CIFs in Figure H.3 reveal distinct risk trajectories. For instance, the blue cluster (estimated probability of death = 0.91) exhibits both a high risk of death and an elevated probability of receiving a liver transplant. In contrast, the green cluster (risk of death = 0.58) is also associated with elevated mortality but primarily at later times, and with substantially lower likeli-

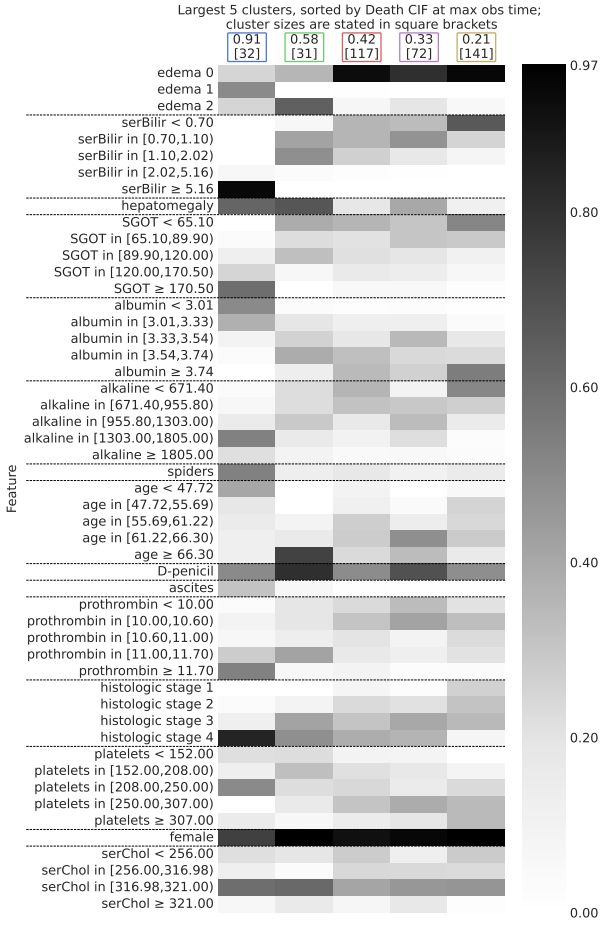

Figure H.4: (PBC) Feature heatmap summarizing distributions of variables in the same 5 clusters as in Figure H.3. Darker shades mean higher feature values or frequencies.

hood of transplant compared to the blue cluster. The heatmap in Figure H.4 highlights the covariate patterns underlying these clusters, showing that clinical characteristics such as edema indicator and histologic stage differentiate high-risk from lower-risk groups.

Turning to the SEER dataset, Figures H.5 and H.6 display the largest 5 clusters of female breast cancer patients diagnosed in 2010. Here, clusters with similar breast cancer mortality (e.g., red and green versus blue) diverge markedly in their competing risk of cardiovascular death. In particular, the blue cluster shows a substantially higher probability of CVD mortality despite having death from breast cancer risk comparable to other clusters. Inspection of the feature heatmap in Figure H.6 reveals that this cluster corresponds to an older patient subgroup, consistent with the increased CVD burden.

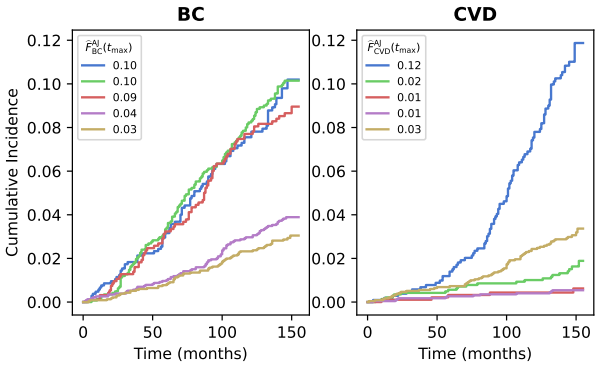

Figure H.5: (SEER) CIFs for the largest 5 clusters (we then sort these 5 clusters in decreasing order by the estimated probability of BC happening within the maximum observed time). Clusters correspond across the two plots in this figure as well as in Figure H.6.

Kernel similarity matrix for Framingham Figure H.7 visualizes the learned kernel from equation (14) after training. Rows/columns are permuted so that training points are grouped by the exemplar-based clusters produced in Step 3 of Section 3.2 (ε-net clustering). We display the 50 largest clusters for the Framingham dataset. The heatmap exhibits a clear block-diagonal pattern: high-intensity diagonal blocks indicate strong within-cluster similarity, while low off-diagonal values reflect separation across clusters; faint bands between blocks suggest clusters that are nearby in the learned embedding.

We point out that this kernel matrix can be used to help understand similarity structure using, for instance, hierarchical clustering (which could show which specific clusters of data points are considered more similar to each other according to the learned similarity/kernel matrix). In particular, many standard hierarchical clustering methods can take as input a similarity/kernel matrix instead of feature vectors (as concrete examples, the agglomerative clustering and spectral clustering methods implemented in scikit-learn (Pedregosa et al., 2011) support this).

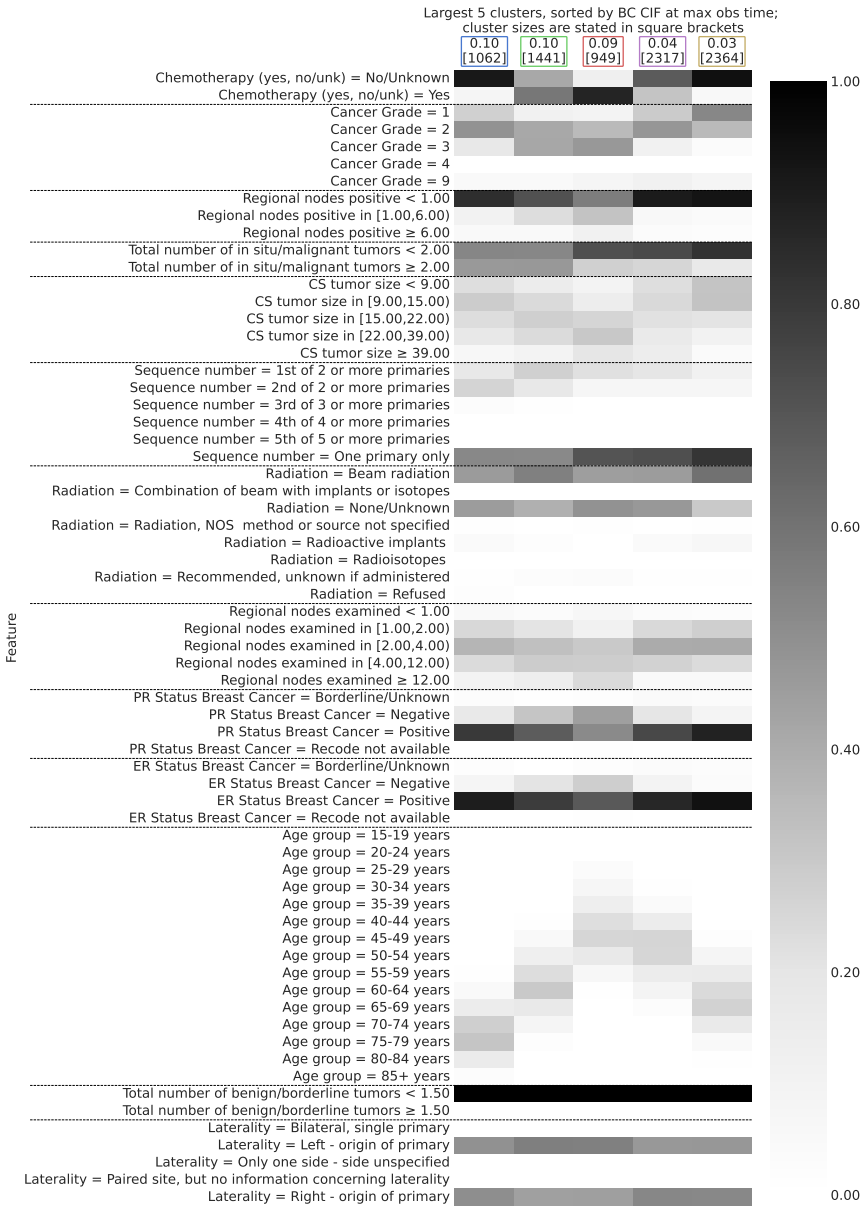

Figure H.6: (SEER) Feature heatmap summarizing distributions of variables in the same 5 clusters as in Figure H.5. Darker shades mean higher feature values or frequencies. We only display a subset of features due to space limit

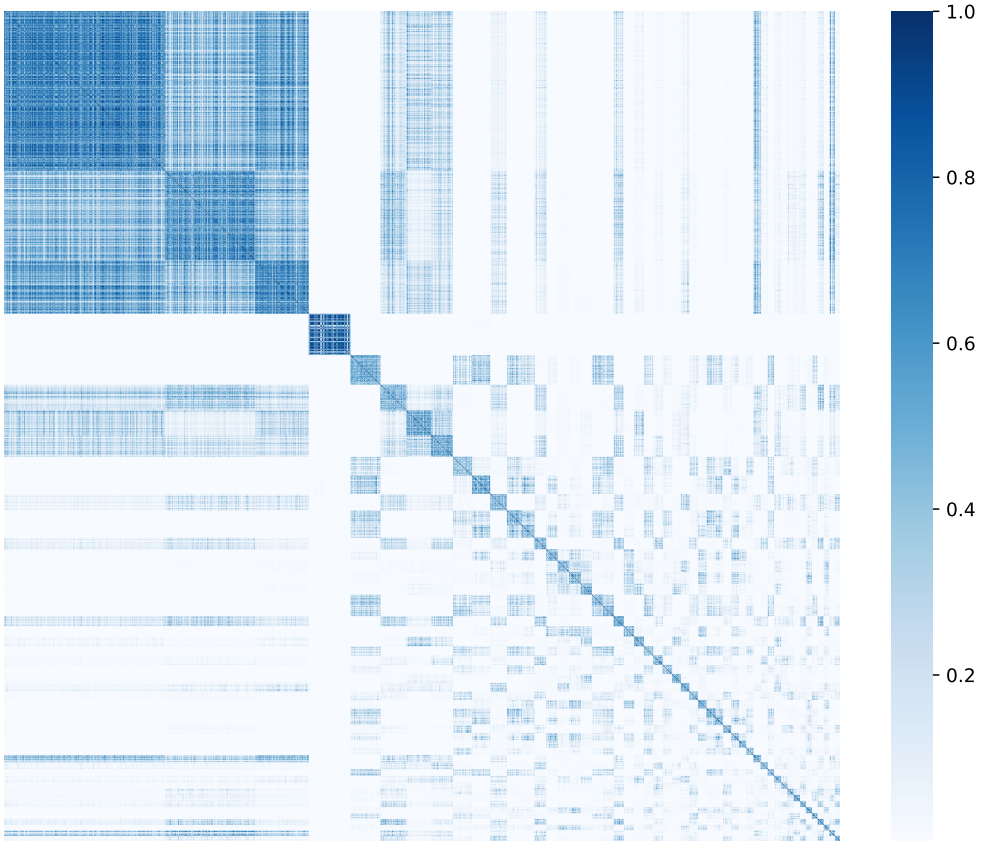

Figure H.7: Kernel similarity matrix $K(X_i, X_j)$ (equation (14)) for the 50 largest DKAJ clusters on the Framingham dataset. Rows/columns are ordered by exemplar-based cluster assignments; darker indicates higher similarity.