A Scalable Nonparametric Continuous-Time Survival Model through Numerical Quadrature

Chaeyeon Lee
Department of Statistics
Ewha Womans University
Seoul, Korea

Sehwan Kim*
Department of Statistics
Ewha Womans University
Seoul, Korea

Hyungrok Do
Department of Population Health
NYU Grossman School of Medicine
New York, NY, USA

Abstract

Flexible continuous-time survival modeling is critical for capturing complex temporal dynamics in high-dimensional data; however, training such models remains challenging due to the intractable integral required for likelihood estimation. We introduce QSurv, a scalable deep learning framework that enables nonparametric continuous-time modeling without relying on time discretization or restrictive distributional assumptions. We propose a training objective based on Gauss-Legendre numerical quadrature, which approximates the cumulative hazard with high-order accuracy while facilitating efficient end-to-end training via standard backpropagation. Furthermore, to effectively capture non-stationary dynamics in complex architectures, we introduce time-conditioned low-rank adaptation, a mechanism that conditions general neural backbones on time by dynamically modulating weights via low-rank updates. We provide theoretical analysis establishing approximation error bounds for cumulative-hazard evaluation. Comprehensive experiments across synthetic benchmarks, large-scale real-world tabular datasets, and high-dimensional medical imaging tasks demonstrate that QSurv achieves competitive predictive performance with advantages in instantaneous hazard function estimation, enabling more interpretable characterization of time-varying risk patterns.

1 Introduction

Survival analysis models the time to an event of interest and is fundamental to medical prognosis, engineering reliability, and financial risk stratification. Its primary challenge lies in handling censoring, where the event time is not fully observed for some subjects within the study period. In this work, we focus on right-censoring, which is the most prevalent form of censoring in practice. Classical methods such as the Kaplan-Meier (KM) estimator [29] and the Cox proportional hazards (PH) model [7] are widely used in survival analysis. The KM estimator serves as a nonparametric approach to estimating the population-level survival function, whereas the Cox PH model provides a semiparametric framework linking covariates to the hazard via a log-linear predictor. While these methods remain standard tools owing to their simplicity and interpretability, they are often insufficient to capture the complex, high-dimensional interactions between covariates and time-to-event. Although machine learning approaches like random survival forests [27] and classification-based frameworks such as survival stacking [8] offer improved risk prediction over classical methods, they remain limited in their ability to directly learn representations from unstructured and high-dimensional data, such as electronic health records and medical imaging, and can impose a substantial computational burden at scale.

*Corresponding Author (sehwankim@ewha.ac.kr)

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To address these limitations, the field has increasingly turned to deep learning. One prominent line of work preserves the Cox PH framework while using a neural network to learn a nonlinear risk function [30]. Another common direction adopts a discrete-time formulation, predicting hazards or event probabilities on a fixed time grid to enable flexible training objectives [33]. Alternatively, parametric deep models support likelihood-based training but constrain the hazard through a chosen distribution family [35]. Notably, these approaches typically rely on structural assumptions about the survival function or hazard. Cox-based neural models primarily learn a relative-risk score under proportional hazards, discrete-time models represent risk on a prespecified time grid, and parametric models restrict the hazard through a chosen distributional family. Thus, their limitations are not only computational or predictive, but also concern how directly they characterize the continuous-time instantaneous hazard.

Direct characterization of the instantaneous hazard is clinically important because the hazard function is the most natural object for describing time-resolved risk. While the survival function summarizes accumulated risk up to time t, the instantaneous hazard $h(t|x)$ describes how risk changes locally among individuals who remain at risk. Many clinical settings exhibit such time-local structure, including delayed treatment effects, transient early toxicity, crossing hazards, late recurrence, and decaying or accelerating risks over follow-up. These patterns are difficult to summarize by a single proportional-hazards effect and may be obscured when models focus only on survival probabilities or fixed-grid event probabilities. Prior work has therefore emphasized both the limitations of routine hazard-ratio reporting under non-proportional hazards [23, 43] and the practical value of examining hazard functions to understand disease dynamics and recurrence patterns [40, 24].

Despite this need, fully nonparametric deep learning for continuous-time hazard modeling remains underexplored because the survival likelihood requires both pointwise hazard evaluation and cumulative-hazard integration. Directly parameterizing $\lambda_\theta(t|x)$ with a flexible time-conditioned network is therefore computationally challenging. Existing deep survival models often avoid this difficulty through proportional-hazards, discrete-time, or parametric assumptions. More flexible continuous-time approaches have recently been proposed, but SODEN [44] depends on adaptive ODE solvers, while DeSurv [10] models the cumulative incidence function rather than the hazard as the primitive quantity. Thus, scalable neural survival modeling that directly parameterizes the instantaneous hazard remains underdeveloped.

Beyond the integration challenge, direct hazard modeling also raises the question of how time should enter the network. Since $h(t|x)$ is a local-in-time risk function, the model must represent interactions between covariates x and follow-up time t, rather than treating time as a passive index. A standard approach is to concatenate t with x at the input, which is often adequate for tabular data processed by multilayer perceptrons; however, for more expressive backbones such as ResNet [21], DenseNet [26] and Transformers [47, 14], simple input concatenation may be insufficient. An alternative is to inject t into the feature representation learned by the backbone, so that time interacts with the covariates at the feature level rather than only at the input.

To address the gap between scalable deep learning and direct hazard modeling, we introduce quadrature-based nonparametric continuous-time deep survival analysis (QSurv). QSurv parameterizes the instantaneous hazard $\lambda_\theta(t|x)$ as a flexible function of time and covariates, and uses numerical quadrature to evaluate the cumulative hazard required by the continuous-time likelihood. In this way, QSurv makes the hazard function the primary modeled object while avoiding the computational burden of adaptive ODE solvers and the precision loss of time discretization. Our main contributions are summarized as follows:

- We propose a nonparametric continuous-time deep survival model that directly parameterizes the instantaneous hazard as a flexible function of time and covariates, enabling time-resolved characterization of risk without fixed time discretization.
- We introduce a likelihood objective based on numerical quadrature. This eliminates the need for ODE solvers, enabling efficient end-to-end training with standard backpropagation and minibatch stochastic gradient descent.
- We propose time-conditioned low-rank adaptation (Time-LoRA), which conditions general neural network backbones on t at the penultimate layer, so that the backbone is evaluated once per subject and reused across all quadrature nodes. Time-LoRA is used throughout our experiments, and FiLM and concatenation at the same layer are viable alternatives.

- We perform comprehensive experiments across synthetic benchmarks, real-world tabular data, and high-dimensional medical imaging tasks. Our results confirm QSurv's efficacy and feasibility compared to state-of-the-art baselines.

2 Related Works

Integrating deep learning approaches into survival analysis has led to a wide range of methodological advances. While these deep learning-based survival analysis methods can be categorized in several ways, to keep the discussion focused, we organize the literature into two broad settings: (1) discrete-time models and (2) continuous-time models. For a comprehensive review, see [48] and the references therein.

In discrete-time survival models, time is partitioned into a set of intervals, and the model outputs interval-specific hazards or a discrete probability distribution over discretized time points. Multi-Task Logistic Regression (MTLR) [49] casts survival estimation as a sequence of binary classification tasks, one for each time interval, using logistic regression. This framework has been extended to neural MTLR (N-MTLR) to capture nonlinear effects [15]. DeepHit [33] similarly models discrete-time event probabilities with a neural network and accommodates competing risks by incorporating an additional ranking-based loss. In contrast, Nnet-survival estimates the conditional hazard probabilities in discrete time points [16].

For continuous-time survival models, existing deep learning approaches can be broadly grouped into three directions.

Cox models: Many methods start from the Cox proportional hazards (PH) framework and relax parts of the original formulation to improve flexibility. A widely used example is DeepSurv, which replaces the linear predictor in the Cox model with a neural network [30]. Several extensions have been proposed. CoxCC modifies the loss to mimic a case–control sampling scheme, and CoxTime allows the hazard to depend on time through a time-varying effect model [32]. Deep Cox Mixtures further extends DeepSurv by modeling heterogeneous subpopulations via a mixture formulation [36]. More generally, the Deep Extended Hazard model provides a flexible hazard formulation that generalizes the baseline hazard structure [50].

Parametric survival models: Another line of work adopts fully parametric specifications, where the survival time is modeled as a function of covariates and error term, typically assuming distributions such as Weibull or log-normal. Neural networks are then used to learn covariate-dependent parameters [39, 4, 1, 35]. On the other hand, Survival Mixture Density Networks model the event-time density using a Gaussian mixture and use a neural network to map covariates to the mixture parameters [20].

ODE-based models: More recently, ODE-based approaches have been proposed that model the evolution of the cumulative hazard (or related quantities) via an ordinary differential equation [44, 45]. This yields a principled continuous-time formulation capable of capturing complex temporal dynamics. A key limitation, however, is that these models can be computationally very expensive to train, and they also introduce additional hyperparameters related to ODE-solver. A related approach, DeSurv [10], formulates an ODE for the cumulative distribution function and discretizes it via Gauss–Legendre quadrature, but recovers the event density via time differentiation rather than directly modeling the hazard.

The literature highlights key limitations that motivate our work. Existing methodologies often impose structural assumptions on the hazard or event-time distribution: discrete-time models depend on a chosen time grid, Cox models inherit restrictions from the PH formulation, and parametric models rely on specific distributional assumptions. While ODE-based formulations offer flexibility in modeling hazard functions, they incur significant computational costs, and the existing quadrature-based alternative bypasses the hazard altogether by modeling the cumulative distribution function. This leaves a clear gap: scalable, fully nonparametric deep learning that directly models the instantaneous hazard remains underexplored. Yet the instantaneous hazard is itself a clinically meaningful quantity, providing a direct measure of risk at any given time that informs time-sensitive decisions such as patient monitoring and intervention timing.

3 Problem Formulation

Let $T, C \in \mathbb{R}_+$ denote the random variables for the true time-to-event and censoring time, respectively. We consider a setting with covariates $X \in \mathcal{X} \subseteq \mathbb{R}^d$. Under the assumption of covariate-dependent censoring, we assume that the event time and censoring time are conditionally independent given the covariates, i.e., $T \perp C|X$. In the observational setting, the true event time T is not always observed directly due to censoring. Instead, we observe the pair (O, Δ), where $O = \min(T, C)$ denotes the observed time and $\Delta = \mathbb{I}(T \leq C)$ is the binary event indicator. Given a dataset of n i.i.d. realizations $\mathcal{D} = \{(x_i, o_i, \delta_i)\}_{i=1}^n$, our primary objective is to learn the parameters θ of the conditional hazard function $\lambda_\theta(t|x)$. More broadly, however, survival analysis aims to characterize the full time-to-event distribution, which may involve estimating the survival function $S_\theta(t|x)$, the cumulative hazard $\Lambda_\theta(t|x)$, or discrete-time event probabilities.

The hazard function $\lambda_\theta(t|x)$ fully characterizes the time-to-event distribution. It is linked to the cumulative hazard $\Lambda_\theta(t|x)$ and the survival function $S_\theta(t|x)$ through the following equations:

$$\Lambda_\theta(t|x) = \int_0^t \lambda_\theta(u|x)du, \qquad S_\theta(t|x) = \exp\left(-\Lambda_\theta(t|x)\right). \tag{1}$$

To estimate the model parameters θ, we minimize the negative log-likelihood (NLL). Under the standard assumption of covariate-dependent censoring ($T \perp C|X$), the terms governing the censoring distribution are independent of θ and factor out of the optimization. This yields the standard survival analysis objective function:

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[\delta_i \log \lambda_\theta(o_i|x_i) - \Lambda_\theta(o_i|x_i)\right]. \tag{2}$$

Minimizing this objective with respect to θ requires a modeling approach that is both expressive enough to capture complex temporal dynamics and computationally tractable for evaluating the cumulative hazard integral. Existing literature navigates this trade-off through various approximations. This objective exposes the central computational difficulty in direct neural network parameterized hazard modeling: the event contribution requires evaluating the hazard locally at o_i, whereas the survival contribution requires integrating the same learned hazard over the full interval $[0, o_i]$. QSurv is designed to preserve the hazard as the primitive modeled quantity while making this cumulative-hazard evaluation efficient and differentiable.

3.1 Quadrature-based Nonparametric Continuous-Time Deep Survival Analysis (QSurv)

We parameterize the hazard using an unconstrained network output $f_\theta : \mathcal{X} \times \mathbb{R}_+ \to \mathbb{R}$ and set $\lambda_\theta(t|x) = \text{softplus}\{f_\theta(x, t)\}$. Minimizing the negative log-likelihood requires evaluating the cumulative hazard $\Lambda_\theta(o_i|x_i) = \int_0^{o_i} \lambda_\theta(s|x_i)\, ds$. While the log-hazard requires only a single forward pass, the cumulative hazard is analytically intractable due to the lack of parametric structure. To resolve this, we approximate $\Lambda_\theta(t|x)$ using K-point Gauss-Legendre quadrature. By applying the change of variables $s = t\tau$ to map the integration domain $[0, t]$ to the canonical interval $[0, 1]$, we obtain the approximation:

$$\hat{\Lambda}_\theta(t|x) = \frac{t}{2}\sum_{k=1}^{K} w_k\, \lambda_\theta(t\tau_k|x), \tag{3}$$

where $\tau_k = (\xi_k + 1)/2$ are the Gauss-Legendre nodes shifted from the standard interval $[-1, 1]$ to $[0, 1]$ via the affine map, and $\{w_k\}_{k=1}^K$ are the corresponding standard Gauss-Legendre weights satisfying $\sum_k w_k = 2$. The factor $t/2$ accounts for the Jacobian of the change of variables $s = t\tau$ combined with the $[-1, 1] \to [0, 1]$ affine transformation.

Unlike standard fixed-grid approximations, such as Riemann sums or the trapezoidal rule, which are constrained to equidistant points, Gauss-Legendre quadrature optimizes node locations to maximize precision. The standard nodes are derived from the roots of the K-th degree Legendre polynomial $P_K(\xi)$ and are mapped to our target domain $[0, 1]$ via the affine transformation $\tau_k = (\xi_k + 1)/2$. The resulting rule is exact for polynomials of degree up to $2K - 1$, and therefore achieves high accuracy for sufficiently smooth hazards with relatively few evaluations. Consequently, the approximation error is small for smooth hazard functions even with a small K. Crucially, this approximation is

Algorithm 1 Training procedure for QSurv

Require: Dataset $\mathcal{D} = \{(x_i, o_i, \delta_i)\}_{i=1}^n$, Gauss-Legendre nodes $\boldsymbol{\tau} \in [0,1]^K$ and weights $\mathbf{w} \in \mathbb{R}^K$.
Initialize network parameters θ.
while not converged **do**
  Sample minibatch $\mathcal{B} \subset \mathcal{D}$ of size B.
  for all $(x_i, o_i, \delta_i) \in \mathcal{B}$ **in parallel do**
    Compute log-hazard by feed-forward: $\log \mathrm{softplus}(f_\theta(x_i, o_i))$
    Estimate cumulative hazard (vectorized): $\hat{\Lambda}_i \leftarrow \frac{o_i}{2} \cdot \mathbf{w}^T \mathrm{softplus}\big(f_\theta(x_i, o_i\boldsymbol{\tau})\big)$.
  end for
  Compute loss: $\mathcal{L}(\theta; \mathcal{B}) \leftarrow -\frac{1}{|\mathcal{B}|} \sum_{(x_i,o_i,\delta_i)\in\mathcal{B}} \Big[\delta_i \log \mathrm{softplus}(f_\theta(x_i, o_i)) - \hat{\Lambda}_i\Big]$.
  Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta; \mathcal{B})$
end while

implemented as a vectorized set of K network evaluations that remains fully differentiable with respect to θ, enabling end-to-end training via standard backpropagation with a time complexity of $\mathcal{O}(K)$ per cumulative hazard evaluation.

Importantly, compared to these Riemann or trapezoidal approximations, Gauss-Legendre quadrature is not tied to a dense time discretization. While grid-based methods require a large number of points to minimize discretization error, scaling computational cost linearly with the desired resolution, Gauss-Legendre leverages optimal node placement to achieve superior accuracy with a fixed, low budget of K evaluations. The parameters of QSurv are learned by minimizing the negative log-likelihood of the observed data. Given a dataset the objective function is defined as:

$$\mathcal{L}(\theta; \mathcal{D}) = -\frac{1}{n} \sum_{i=1}^{n} \Big[\delta_i \log \mathrm{softplus}(f_\theta(x_i, o_i)) - \frac{o_i}{2} \sum_{k=1}^{K} w_k \mathrm{softplus}(f_\theta(x_i, o_i\tau_k))\Big], \tag{4}$$

where the first term captures the instantaneous risk for observed events, and the second term is the cumulative hazard, approximated via the quadrature method described above. Algorithm 1 describes the training procedure for QSurv.

3.2 Approximation Error Bound

The validity of QSurv relies on the convergence of the quadrature approximation to the exact cumulative-hazard integral induced by the modeled hazard. This follows from classical properties of Gauss-Legendre quadrature: the K-point rule is exact for polynomials of degree up to $2K-1$ and admits a high-order error bound for sufficiently smooth integrands [17, 11]. In our setting, the integrand is the neural network parameterized hazard $\lambda_\theta(t|x)$, so the quadrature error is controlled by the temporal smoothness of the learned hazard. We state the resulting bound below and defer the proof, which is a direct application of the standard Gauss-Legendre error formula after an affine transformation from $[-1, 1]$ to $[0, t]$, to Appendix B.

Theorem 3.1 (Approximation Error Bound of Cumulative Hazard Function)**.** *Let the hazard function $\lambda_\theta(\cdot|x)$ be $2K$-times continuously differentiable on $[0, t_{\max}]$. Let $\Lambda_\theta(t|x)$ denote the exact cumulative hazard induced by λ_θ and $\hat{\Lambda}_{\theta,K}(t|x)$ be the approximation computed using K-point Gauss-Legendre quadrature. Then, for any $t \in [0, t_{\max}]$, the approximation error is bounded by:*

$$\varepsilon_K(t|x) = \Big|\Lambda_\theta(t|x) - \hat{\Lambda}_{\theta,K}(t|x)\Big| \le \frac{t^{2K+1}(K!)^4}{(2K+1)[(2K)!]^3} \max_{\tau\in[0,t]} \Big|\lambda_\theta^{(2K)}(\tau|x)\Big|. \tag{5}$$

Furthermore, if $\lambda_\theta(\cdot|x)$ is a polynomial of degree at most $2K-1$ with respect to time, the approximation is exact, i.e., $\hat{\Lambda}_{\theta,K}(t|x) = \Lambda_\theta(t|x)$.

Theorem 3.1 shows that the quadrature error is controlled by the number of nodes K and the temporal smoothness of the learned hazard. For sufficiently smooth hazards, the factorial term in the bound yields rapid error decay as K increases, and the Gauss–Legendre rule is exact for hazards that are polynomial in time of degree at most $2K-1$. Thus, a moderate fixed value of K can provide

accurate cumulative-hazard estimates in practice, as confirmed empirically in Appendix C. The result also motivates using smooth hazard parameterizations, such as networks with Softplus or GELU activations, since high-order quadrature accuracy relies on the existence of temporal derivatives. Overall, K should be viewed as a numerical precision parameter rather than a model-complexity or regularization parameter.

We establish a direct link between the numerical precision of the integration scheme and the training objective. Specifically, we show that the error in the negative log-likelihood is bounded by the error in the cumulative hazard approximation.

Corollary 3.2 (Propagation of Approximation Error)**.** *Let $\mathcal{L}(\theta; t, \delta, x) = -\delta \log \lambda_\theta(t|x) + \Lambda_\theta(t|x)$ be the true negative log-likelihood for a sample with time t, event indicator δ, and features x. Let $\hat{\mathcal{L}}_K$ denote the approximate loss computed using the quadrature estimate $\hat{\Lambda}_{\theta,K}$. Then the loss approximation error equals the cumulative hazard approximation error:*

$$\left|\mathcal{L}(\theta; t, \delta, x) - \hat{\mathcal{L}}_K(\theta; t, \delta, x)\right| = \varepsilon_K(t|x). \tag{6}$$

This result relates the stability of the learning process to the integration accuracy. Because the cumulative hazard enters the negative log-likelihood linearly, the approximation error is additive and does not compound through non-linear transformations. Consequently, the bound $\varepsilon_K(t|x)$ translates directly to the loss function. When combined with the convergence properties established in Theorem 3.1, this suggests that controlling the quadrature error effectively limits the deviation of the training objective from the true likelihood, allowing the approximate loss to serve as a reliable proxy during optimization.

3.3 Choice of the Number of Quadrature Nodes

The number of quadrature nodes K controls the numerical accuracy of the cumulative-hazard approximation and the computational cost of training. Importantly, K is not a time-discretization parameter in the same sense as the number of bins in discrete-time survival models. QSurv still represents the hazard as a continuous function of time; K only determines how accurately the integral of this hazard is approximated inside the likelihood. Very small values of K may provide an inaccurate approximation when the hazard varies rapidly over time, which can distort the training objective. Increasing K improves the fidelity of the quadrature approximation but requires additional evaluations of the neural network parameterized hazard at the quadrature nodes. Thus, K should be interpreted as a numerical precision parameter rather than a regularization hyperparameter: reducing K does not impose a simpler hazard model, but instead coarsens the approximation to the cumulative hazard. In practice, we use a moderate fixed value of K and examine sensitivity to this choice empirically in Appendix C.

3.4 Time-Conditioned LoRA

For tabular data, a common strategy to model time-varying hazards is to treat time t as an additional input feature, simply concatenating it with the static covariates x, e.g., as in CoxTime [32], DeSurv [10], and SODEN [44]. However, this approach does not trivially extend to high-dimensional data such as medical images processed by backbones like ResNet [21], DenseNet [26] and Transformers [47, 14]. Concatenating a scalar t directly to an image input is not naturally compatible with convolutional or transformer backbones. To address this, we introduce time-conditioned LoRA, which uses a time-dependent weight matrix at the network's penultimate layer to make the hazard prediction adaptive in t without re-running the backbone.

For this penultimate layer with weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and bias $b \in \mathbb{R}^{d_{\text{out}}}$, we parameterize the time-varying weight $W(t)$ as a low-rank perturbation of the static base [25]:

$$W(t) = W + \Delta W(t) = W + U \cdot \text{diag}(s(t)) \cdot V, \tag{7}$$

where $U \in \mathbb{R}^{d_{\text{out}} \times r}$ and $V \in \mathbb{R}^{r \times d_{\text{in}}}$ are learnable low-rank projection matrices ($r \ll \min(d_{\text{in}}, d_{\text{out}})$). The temporal dynamics are captured by the modulation coefficients $s(t) = g_\psi(\phi(t)) \in \mathbb{R}^r$, which are generated by a lightweight MLP g_ψ acting on a time embedding $\phi(t)$ [31].

The forward pass can be re-interpreted as a time-gated residual branch operating within a learned subspace:

$$z(t) = Wh + U\big(s(t) \odot (Vh)\big) + b. \tag{8}$$

Table 1: Comparison of L_1 error of instantaneous hazard (lower the better) across different parametric distributions with standard deviations over 20 random seeds. Best results are **bold**, second best are underlined. Full results are presented in Table 5 in the appendix.

Distribution	CoxCC	CoxTime	NnetSurv	MDN	DeSurv	SODEN	QSurv
Exponential	0.1775 ± 0.0410	0.1775 ± 0.0422	0.3830 ± 0.0438	0.0568 ± 0.0446	0.0448 ± 0.0159	$\mathbf{0.0330 \pm 0.0138}$	0.0352 ± 0.0164
Weibull	0.1576 ± 0.0322	0.1182 ± 0.0207	0.2383 ± 0.0254	0.0644 ± 0.0102	0.0295 ± 0.0090	$\mathbf{0.0272 \pm 0.0119}$	0.0282 ± 0.0139
Gamma	0.9012 ± 0.0921	0.5661 ± 0.1769	0.7249 ± 0.0370	0.5646 ± 0.0459	0.2930 ± 0.0907	$\mathbf{0.1716 \pm 0.0901}$	0.1718 ± 0.0859
Gompertz	0.1713 ± 0.0262	0.1890 ± 0.0549	0.2921 ± 0.0276	0.0920 ± 0.0140	0.0502 ± 0.0278	0.0262 ± 0.0174	$\mathbf{0.0258 \pm 0.0180}$
Log-normal	0.1235 ± 0.0248	0.1114 ± 0.0294	0.1938 ± 0.0219	0.1882 ± 0.0520	0.0473 ± 0.0201	0.0462 ± 0.0183	$\mathbf{0.0457 \pm 0.0185}$
Log-logistic	0.1621 ± 0.0210	0.1266 ± 0.0242	0.3290 ± 0.0525	0.1031 ± 0.0090	0.0513 ± 0.0299	0.0517 ± 0.0295	$\mathbf{0.0491 \pm 0.0291}$

Here, V projects the input h into a latent subspace, $s(t)$ applies a time-varying activation, and U projects the result back to the output dimension; a final time-independent linear layer maps $z(t)$ to an unconstrained scalar $f_\theta(x,t)$, which is transformed into the hazard through $\lambda_\theta(t|x) = \mathrm{softplus}(f_\theta(x,t))$. This structural decoupling yields a significant computational advantage during numerical integration, where the hazard must be evaluated at multiple quadrature nodes $\{\tau_k\}_{k=1}^K$ for a single subject. Because Time-LoRA is applied only at the penultimate layer, the embedding h is independent of t, so the base computation Wh and the initial projection Vh depend solely on the static input x and are computed only once and cached. For each quadrature node τ_k, only the lightweight temporal branch is evaluated to obtain $s(o_i\tau_k)$, followed by the low-rank recombination. This avoids K redundant forward passes of the heavy backbone, drastically reducing the computational cost compared to fully time-entangled architectures. This caching is only valid because the modulation is confined to the head: applying Time-LoRA to an intermediate backbone layer would make the inputs to all downstream layers time-dependent and require a full backbone pass per quadrature node, which is why we restrict it to the penultimate layer. Unlike FiLM, which applies element-wise affine modulation [38], Time-LoRA allows time-dependent low-rank mixing across feature channels. The computational benefit comes from where time enters rather than from the mechanism: FiLM and concatenation applied at the same penultimate layer share the caching property. We use Time-LoRA throughout and treat it as one viable conditioning choice. The simulation ablation in Appendix E and the real-data comparison in Table 6 compare the three mechanisms.

4 Experiments

We performed a series of experiments across a comprehensive set of benchmark datasets to evaluate the proposed methods. To ensure a rigorous comparison across different modeling paradigms, we evaluated against Cox-style deep learning models CoxCC and CoxTime [32], discrete-time models DeepHit [33] and NnetSurv [16], and continuous-time estimators including Survival MDN [20], DeSurv [10], and SODEN [44]. For a controlled comparison of time-conditioned continuous-time architectures, CoxTime, DeSurv, and SODEN were implemented with the same time-conditioned LoRA mechanism used in QSurv. Details of the deep survival models are provided in Appendix F.7.

4.1 Simulation Study

We conducted a simulation study to assess whether QSurv can recover instantaneous hazard functions under nonlinear covariate effects. Details of the data-generating mechanisms and additional results are provided in Appendix D. Table 1 reports L_1 error on the test set for the estimated instantaneous hazard functions. QSurv and SODEN consistently produced the smallest hazard-estimation errors, particularly for distributions with more complex hazard shapes. The main difference was computational: QSurv achieved similar accuracy with much shorter training time. Across simulation settings, SODEN required approximately 130 seconds on average, compared with approximately 6 seconds for QSurv. Figures 6–11 show the corresponding test-set estimates of the instantaneous hazard, cumulative hazard, and survival functions.

4.2 Real-world Datasets

We evaluated QSurv on nine benchmark datasets, comprising six tabular datasets (FLCHAIN, FRAMINGHAM, GBSG, METABRIC, NWTCO, SUPPORT2) and three radiology imaging datasets (COVID-19-NY, C4KC-KiTS, and BraTS). These datasets cover diverse medical domains and vary substantially in sample size, feature dimensionality, and censoring rates; their main characteristics are

Table 2: Performance of QSurv vs deep survival baselines across the full and quantile-based horizons. Best results are **bold**, second best are underlined. The right-most D-cal column is the mean p-value of the discrete-time D-calibration goodness-of-fit test (higher is better; $p > 0.05$ indicates no calibration violation).

		Horizon: Full ($\hat{G}(\tau)$=0.001)			**Horizon:** T_{Q_1}			**Horizon:** T_{Q_2} **(median)**			
Dataset	Model	C_{td} ↑	IBS ↓	IBLL ↑	C_{td} ↑	IBS ↓	IBLL ↑	C_{td} ↑	IBS ↓	IBLL ↑	D-cal ↑
COVID-19-NY	CoxCC	0.5693 ± 0.0961	0.1885 ± 0.0882	-0.5211 ± 0.2118	0.5925 ± 0.1640	0.0048 ± 0.0052	-0.0309 ± 0.0349	0.5504 ± 0.0633	0.0366 ± 0.0090	-0.1656 ± 0.0339	0.7264 ± 0.4338
	CoxTime	0.5587 ± 0.0847	0.2079 ± 0.1211	-0.6197 ± 0.3588	0.5901 ± 0.1031	**0.0044 ± 0.0046**	-0.0306 ± 0.0337	0.5636 ± 0.0891	0.0334 ± 0.0100	-0.1592 ± 0.0327	0.6061 ± 0.5335
	DeepHit	0.6677 ± 0.0793	0.0708 ± 0.0209	-0.2426 ± 0.0600	0.4951 ± 0.1224	0.0045 ± 0.0046	-0.0291 ± 0.0282	0.6516 ± 0.0859	0.0371 ± 0.0130	-0.1660 ± 0.0491	0.7470 ± 0.3440
	NnetSurv	0.6660 ± 0.0844	0.0797 ± 0.0159	-0.2629 ± 0.0406	0.5753 ± 0.0990	0.0045 ± 0.0046	**-0.0265 ± 0.0259**	0.6793 ± 0.1175	0.0344 ± 0.0069	-0.1463 ± 0.0171	0.8787 ± 0.2473
	MDN	0.6889 ± 0.0626	0.1892 ± 0.0797	-0.5626 ± 0.2426	0.5669 ± 0.1987	0.0185 ± 0.0100	-0.0954 ± 0.0358	0.7116 ± 0.0536	0.0406 ± 0.0107	-0.1660 ± 0.0323	0.7658 ± 0.2906
	DeSurv	0.7420 ± 0.0923	**0.0617 ± 0.0269**	**-0.1981 ± 0.0802**	0.7173 ± 0.2020	0.0052 ± 0.0049	-0.0367 ± 0.0258	0.7944 ± 0.0615	**0.0322 ± 0.0089**	**-0.1410 ± 0.0334**	0.8502 ± 0.1983
	SODEN	0.7228 ± 0.0803	0.0736 ± 0.0190	-0.2886 ± 0.1269	0.6271 ± 0.2349	0.0050 ± 0.0047	-0.0341 ± 0.0231	0.7618 ± 0.1001	0.0332 ± 0.0076	-0.1413 ± 0.0211	0.8653 ± 0.2780
	QSurv	**0.7456 ± 0.1046**	0.0716 ± 0.0155	-0.2318 ± 0.0489	**0.7260 ± 0.1002**	0.0062 ± 0.0050	-0.0405 ± 0.0259	**0.7951 ± 0.0908**	0.0352 ± 0.0157	-0.1441 ± 0.0488	0.7391 ± 0.2436
C4KC-KiTS	CoxCC	0.4116 ± 0.1798	0.1099 ± 0.0050	-0.3644 ± 0.0240	0.4834 ± 0.2542	0.0307 ± 0.0196	-0.1523 ± 0.0805	0.4030 ± 0.1679	0.0740 ± 0.0132	-0.2840 ± 0.0468	1.0000 ± 0.0001
	CoxTime	0.5808 ± 0.1985	**0.1086 ± 0.0015**	**-0.3571 ± 0.0139**	0.5877 ± 0.2812	0.0304 ± 0.0194	-0.1479 ± 0.0788	0.5806 ± 0.1851	**0.0734 ± 0.0124**	**-0.2787 ± 0.0408**	0.9999 ± 0.0001
	DeepHit	0.5090 ± 0.1089	0.2179 ± 0.1236	-0.7631 ± 0.3839	0.5290 ± 0.0644	0.0310 ± 0.0195	-0.1570 ± 0.0838	0.4900 ± 0.1264	0.0820 ± 0.0192	-0.3271 ± 0.0690	0.6826 ± 0.4021
	NnetSurv	0.5422 ± 0.1375	0.1262 ± 0.0240	-0.5230 ± 0.1907	0.7345 ± 0.1797	0.0340 ± 0.0172	-0.1964 ± 0.1066	0.5371 ± 0.1172	0.0867 ± 0.0277	-0.4008 ± 0.1529	0.9574 ± 0.0529
	MDN	0.5230 ± 0.2206	0.1106 ± 0.0139	-0.4742 ± 0.2106	0.5290 ± 0.1920	0.0352 ± 0.0186	-0.1908 ± 0.1246	0.5270 ± 0.2244	0.0773 ± 0.0155	-0.3829 ± 0.2099	0.9693 ± 0.0473
	DeSurv	0.5740 ± 0.1630	0.1156 ± 0.0152	-0.6108 ± 0.3646	0.6274 ± 0.2445	0.0329 ± 0.0232	-0.2136 ± 0.1838	0.5685 ± 0.1636	0.0826 ± 0.0228	-0.4910 ± 0.2947	0.9889 ± 0.0225
	SODEN	0.5573 ± 0.1640	0.1352 ± 0.0253	-1.3282 ± 0.9974	0.6037 ± 0.2447	0.0315 ± 0.0195	-0.2472 ± 0.2239	0.5615 ± 0.1482	0.0841 ± 0.0090	-0.8011 ± 0.4213	0.9729 ± 0.0147
	QSurv	**0.6381 ± 0.1446**	0.1260 ± 0.0453	-0.4002 ± 0.1113	**0.7404 ± 0.1547**	**0.0288 ± 0.0178**	**-0.1220 ± 0.0679**	**0.6207 ± 0.1596**	0.0776 ± 0.0235	-0.2910 ± 0.0644	0.9845 ± 0.0209
BraTS	CoxCC	0.4925 ± 0.0344	0.2021 ± 0.1203	-0.6222 ± 0.3545	0.4976 ± 0.0676	0.1414 ± 0.0227	-0.5501 ± 0.2447	0.4949 ± 0.0382	0.2168 ± 0.0411	-0.7373 ± 0.3262	0.0702 ± 0.1141
	CoxTime	0.4541 ± 0.0359	0.1171 ± 0.0130	-0.3764 ± 0.0363	0.4414 ± 0.0549	0.1355 ± 0.0128	-0.4456 ± 0.0388	0.4460 ± 0.0406	0.2021 ± 0.0127	-0.5939 ± 0.0319	0.3090 ± 0.3068
	DeepHit	0.5196 ± 0.0383	0.1164 ± 0.0175	-0.3765 ± 0.0453	0.5020 ± 0.0679	0.1357 ± 0.0125	-0.4582 ± 0.0378	0.5130 ± 0.0457	0.2010 ± 0.0160	-0.6005 ± 0.0461	0.1414 ± 0.1793
	NnetSurv	0.5355 ± 0.0395	0.1243 ± 0.0394	-0.5086 ± 0.3472	0.5371 ± 0.0533	0.1806 ± 0.1093	-0.5546 ± 0.2757	0.5471 ± 0.0465	0.2599 ± 0.1342	-0.8321 ± 0.5395	0.3879 ± 0.5029
	MDN	0.5333 ± 0.0537	0.1274 ± 0.0103	-0.4131 ± 0.0252	0.5321 ± 0.0342	0.1754 ± 0.0082	-0.5387 ± 0.0175	0.5429 ± 0.0479	0.2120 ± 0.0088	-0.6149 ± 0.0186	0.0000 ± 0.0000
	DeSurv	0.5459 ± 0.0297	**0.1038 ± 0.0176**	-0.3405 ± 0.0707	0.5570 ± 0.0372	**0.1355 ± 0.0141**	**-0.4378 ± 0.0363**	0.5438 ± 0.0297	0.2044 ± 0.0195	-0.5930 ± 0.0451	0.1535 ± 0.1623
	SODEN	0.5505 ± 0.0295	0.1055 ± 0.0189	-0.3508 ± 0.0845	0.5337 ± 0.0652	0.1387 ± 0.0124	-0.4445 ± 0.0289	0.5393 ± 0.0462	0.2013 ± 0.0096	-0.5877 ± 0.0232	0.1573 ± 0.1317
	QSurv	**0.5811 ± 0.0387**	0.1038 ± 0.0133	**-0.3327 ± 0.0371**	**0.6149 ± 0.0559**	0.1380 ± 0.0254	-0.4393 ± 0.0628	**0.5968 ± 0.0537**	**0.1993 ± 0.0329**	**-0.5868 ± 0.0868**	0.2161 ± 0.2395
Average Rank (Image)	CoxCC	7.33	5.33	5.33	6.33	4.33	4.67	7.67	5.00	5.00	
	CoxTime	6.00	4.67	4.33	6.00	**1.50**	**3.00**	5.67	**2.67**	3.33	
	DeepHit	6.00	4.67	5.00	6.83	3.17	3.67	6.33	4.67	5.50	
	NnetSurv	5.00	5.67	5.33	3.67	5.83	5.00	4.00	6.67	6.00	
	MDN	5.00	5.67	5.67	6.17	7.67	6.33	4.67	5.67	6.17	
	DeSurv	2.67	**2.17**	3.00	2.33	4.50	4.67	2.67	4.00	3.67	
	SODEN	3.00	4.67	5.33	3.67	5.00	5.33	4.00	4.00	4.00	
	QSurv	**1.00**	3.17	**2.00**	**1.00**	4.00	3.33	**1.00**	3.33	**2.33**	
FLCHAIN	CoxCC	0.7967 ± 0.0101	0.0936 ± 0.0024	-0.3103 ± 0.0075	0.8024 ± 0.0120	0.0680 ± 0.0023	-0.2395 ± 0.0075	0.7988 ± 0.0099	0.0861 ± 0.0023	-0.2914 ± 0.0076	0.9224 ± 0.1189
	CoxTime	0.7968 ± 0.0101	0.0936 ± 0.0024	-0.3100 ± 0.0074	0.8027 ± 0.0125	0.0680 ± 0.0023	-0.2389 ± 0.0076	0.7989 ± 0.0102	0.0862 ± 0.0023	-0.2910 ± 0.0074	0.9354 ± 0.1325
	DeepHit	0.7883 ± 0.0107	0.1056 ± 0.0026	-0.3400 ± 0.0069	0.7891 ± 0.0145	0.0755 ± 0.0029	-0.2587 ± 0.0081	0.7896 ± 0.0114	0.0974 ± 0.0028	-0.3190 ± 0.0076	0.0192 ± 0.0482
	NnetSurv	0.7887 ± 0.0113	0.0935 ± 0.0028	-0.3103 ± 0.0086	0.7898 ± 0.0152	0.0681 ± 0.0026	-0.2402 ± 0.0082	0.7903 ± 0.0121	0.0864 ± 0.0026	-0.2922 ± 0.0084	0.7866 ± 0.2870
	MDN	0.7954 ± 0.0096	0.0946 ± 0.0020	-0.3142 ± 0.0068	0.8012 ± 0.0117	0.0689 ± 0.0021	-0.2425 ± 0.0069	0.7973 ± 0.0097	0.0872 ± 0.0020	-0.2949 ± 0.0071	0.4600 ± 0.2720
	DeSurv	0.7968 ± 0.0097	0.0936 ± 0.0021	-0.3106 ± 0.0067	0.8030 ± 0.0116	0.0680 ± 0.0023	-0.2398 ± 0.0073	0.7988 ± 0.0097	0.0862 ± 0.0022	-0.2919 ± 0.0070	0.8706 ± 0.1812
	QSurv	**0.7968 ± 0.0096**	**0.0935 ± 0.0023**	**-0.3097 ± 0.0070**	**0.8031 ± 0.0118**	**0.0678 ± 0.0023**	**-0.2387 ± 0.0077**	**0.7989 ± 0.0094**	**0.0860 ± 0.0022**	**-0.2907 ± 0.0073**	0.8819 ± 0.1852
FRAMINGHAM	CoxCC	0.7407 ± 0.0142	0.0882 ± 0.0029	-0.2929 ± 0.0088	0.7556 ± 0.0131	**0.0607 ± 0.0032**	-0.2162 ± 0.0099	0.7407 ± 0.0142	0.0882 ± 0.0029	-0.2929 ± 0.0088	0.9139 ± 0.1385
	CoxTime	0.7411 ± 0.0156	**0.0881 ± 0.0031**	**-0.2925 ± 0.0095**	0.7561 ± 0.0146	0.0608 ± 0.0034	**-0.2160 ± 0.0109**	0.7411 ± 0.0156	**0.0881 ± 0.0031**	**-0.2925 ± 0.0095**	0.9543 ± 0.0693
	DeepHit	0.7262 ± 0.0136	0.1780 ± 0.0162	-0.5220 ± 0.0412	0.7403 ± 0.0136	0.0819 ± 0.0089	-0.2948 ± 0.0254	0.7262 ± 0.0136	0.1780 ± 0.0162	-0.5220 ± 0.0412	0.0000 ± 0.0000
	NnetSurv	0.7316 ± 0.0165	0.0902 ± 0.0033	-0.2997 ± 0.0110	0.7439 ± 0.0159	0.0615 ± 0.0031	-0.2200 ± 0.0103	0.7316 ± 0.0165	0.0902 ± 0.0033	-0.2997 ± 0.0110	0.6171 ± 0.3546
	MDN	0.7378 ± 0.0128	0.0890 ± 0.0025	-0.2960 ± 0.0084	0.7541 ± 0.0125	0.0611 ± 0.0031	-0.2183 ± 0.0108	0.7378 ± 0.0128	0.0890 ± 0.0025	-0.2960 ± 0.0084	0.8712 ± 0.2163
	DeSurv	0.7400 ± 0.0138	0.0887 ± 0.0030	-0.2954 ± 0.0104	0.7565 ± 0.0137	0.0610 ± 0.0035	-0.2178 ± 0.0116	0.7400 ± 0.0138	0.0887 ± 0.0030	-0.2954 ± 0.0104	0.8984 ± 0.1424
	QSurv	**0.7414 ± 0.0146**	0.0884 ± 0.0029	-0.2933 ± 0.0087	**0.7578 ± 0.0145**	0.0607 ± 0.0033	-0.2161 ± 0.0105	**0.7414 ± 0.0146**	0.0884 ± 0.0029	-0.2933 ± 0.0087	0.9004 ± 0.1399
GBSG	CoxCC	0.6554 ± 0.0294	**0.1759 ± 0.0117**	**-0.5220 ± 0.0328**	0.6697 ± 0.0430	0.0866 ± 0.0088	-0.2991 ± 0.0284	0.6678 ± 0.0291	0.1408 ± 0.0139	-0.4386 ± 0.0366	0.7995 ± 0.2822
	CoxTime	0.6495 ± 0.0294	0.1822 ± 0.0176	-0.5385 ± 0.0464	0.6683 ± 0.0425	0.0877 ± 0.0084	-0.3027 ± 0.0258	0.6656 ± 0.0293	0.1433 ± 0.0142	-0.4473 ± 0.0371	0.7181 ± 0.3025
	DeepHit	0.6336 ± 0.0280	0.2071 ± 0.0301	-0.5975 ± 0.0793	0.6497 ± 0.0401	0.0903 ± 0.0087	-0.3162 ± 0.0270	0.6531 ± 0.0314	0.1556 ± 0.0185	-0.4759 ± 0.0436	0.4138 ± 0.4239
	NnetSurv	0.6245 ± 0.0314	0.2001 ± 0.0782	-0.7529 ± 0.9511	0.6401 ± 0.0423	0.1172 ± 0.1302	-0.4358 ± 0.5667	0.6388 ± 0.0301	0.1724 ± 0.1255	-0.6716 ± 0.9790	0.6485 ± 0.3673
	MDN	0.6529 ± 0.0286	0.1818 ± 0.0108	-0.5406 ± 0.0321	0.6666 ± 0.0484	0.0916 ± 0.0093	-0.3187 ± 0.0309	0.6649 ± 0.0286	0.1442 ± 0.0124	-0.4537 ± 0.0334	0.4328 ± 0.3027
	DeSurv	**0.6592 ± 0.0245**	0.1769 ± 0.0120	-0.5248 ± 0.0344	**0.6784 ± 0.0364**	0.0873 ± 0.0074	-0.3029 ± 0.0203	**0.6727 ± 0.0252**	0.1403 ± 0.0113	-0.4378 ± 0.0280	0.3674 ± 0.2050
	QSurv	0.6558 ± 0.0297	0.1780 ± 0.0129	-0.5247 ± 0.0353	0.6765 ± 0.0453	**0.0865 ± 0.0076**	**-0.2983 ± 0.0219**	0.6697 ± 0.0299	**0.1402 ± 0.0134**	**-0.4369 ± 0.0343**	0.6001 ± 0.3362
METABRIC	CoxCC	0.6346 ± 0.0200	0.1549 ± 0.0099	-0.4679 ± 0.0294	0.6229 ± 0.0322	0.1055 ± 0.0050	-0.3561 ± 0.0157	0.6368 ± 0.0260	0.1562 ± 0.0055	-0.4798 ± 0.0158	0.6223 ± 0.2984
	CoxTime	0.6565 ± 0.0229	0.1578 ± 0.0102	-0.4739 ± 0.0304	**0.6739 ± 0.0320**	0.1032 ± 0.0063	-0.3460 ± 0.0230	**0.6637 ± 0.0246**	0.1533 ± 0.0070	-0.4707 ± 0.0244	0.5473 ± 0.3588
	DeepHit	**0.6626 ± 0.0182**	0.1609 ± 0.0102	-0.4844 ± 0.0281	0.6669 ± 0.0300	0.1040 ± 0.0046	-0.3494 ± 0.0131	0.6608 ± 0.0232	0.1571 ± 0.0039	-0.4803 ± 0.0100	0.5653 ± 0.2963
	NnetSurv	0.6459 ± 0.0232	0.1626 ± 0.0263	-0.4870 ± 0.0263	0.6531 ± 0.0312	0.1044 ± 0.0055	-0.3505 ± 0.0170	0.6505 ± 0.0259	0.1565 ± 0.0058	-0.4787 ± 0.0164	0.3262 ± 0.3765
	MDN	0.6365 ± 0.0178	0.1708 ± 0.0089	-0.5143 ± 0.0231	0.6309 ± 0.0334	0.1147 ± 0.0052	-0.3852 ± 0.0142	0.6413 ± 0.0223	0.1600 ± 0.0048	-0.4911 ± 0.0129	0.0000 ± 0.0000
	DeSurv	0.6551 ± 0.0207	0.1554 ± 0.0110	-0.4656 ± 0.0302	0.6660 ± 0.0343	0.1033 ± 0.0039	-0.3460 ± 0.0123	0.6599 ± 0.0261	0.1533 ± 0.0052	-0.4699 ± 0.0140	0.3543 ± 0.2764
	QSurv	0.6542 ± 0.0198	**0.1548 ± 0.0100**	**-0.4638 ± 0.0290**	0.6683 ± 0.0313	**0.1031 ± 0.0049**	**-0.3451 ± 0.0153**	0.6602 ± 0.0220	**0.1531 ± 0.0048**	**-0.4687 ± 0.0137**	0.3580 ± 0.2688
NWTCO	CoxCC	0.6422 ± 0.0345	0.1126 ± 0.0039	-0.3688 ± 0.0141	0.6343 ± 0.0347	0.0800 ± 0.0031	-0.2871 ± 0.0107	0.6394 ± 0.0335	0.1023 ± 0.0029	-0.3503 ± 0.0102	0.9777 ± 0.0471
	CoxTime	0.6377 ± 0.0347	0.1139 ± 0.0047	-0.3740 ± 0.0198	0.6310 ± 0.0346	0.0810 ± 0.0038	-0.2904 ± 0.0136	0.6350 ± 0.0342	0.1036 ± 0.0038	-0.3549 ± 0.0156	0.9341 ± 0.2133
	DeepHit	0.6360 ± 0.0266	0.1147 ± 0.0023	-0.3744 ± 0.0108	0.6282 ± 0.0263	0.0807 ± 0.0025	-0.2891 ± 0.0078	0.6342 ± 0.0259	0.1033 ± 0.0018	-0.3527 ± 0.0077	0.8889 ± 0.2197
	NnetSurv	0.6301 ± 0.0199	0.1155 ± 0.0027	-0.3771 ± 0.0113	0.6236 ± 0.0209	0.0817 ± 0.0030	-0.2930 ± 0.0080	0.6284 ± 0.0197	0.1046 ± 0.0023	-0.3570 ± 0.0070	0.7297 ± 0.3042
	MDN	**0.6532 ± 0.0273**	**0.1102 ± 0.0032**	**-0.3622 ± 0.0117**	**0.6448 ± 0.0269**	0.0808 ± 0.0028	-0.2938 ± 0.0089	**0.6498 ± 0.0263**	0.1032 ± 0.0019	-0.3565 ± 0.0080	0.9426 ± 0.0962
	DeSurv	0.6492 ± 0.0236	0.1120 ± 0.0030	-0.3664 ± 0.0115	0.6398 ± 0.0227	0.0798 ± 0.0030	-0.2862 ± 0.0088	0.6459 ± 0.0231	0.1020 ± 0.0028	-0.3489 ± 0.0095	0.9934 ± 0.0155
	QSurv	0.6480 ± 0.0209	0.1118 ± 0.0027	-0.3667 ± 0.0112	0.6398 ± 0.0204	**0.0793 ± 0.0027**	**-0.2848 ± 0.0075**	0.6449 ± 0.0202	**0.1015 ± 0.0021**	**-0.3477 ± 0.0075**	0.9401 ± 0.1839
SUPPORT2	CoxCC	0.6770 ± 0.0061	0.1798 ± 0.0046	-0.5449 ± 0.0201	0.7122 ± 0.0079	0.1202 ± 0.0022	-0.3896 ± 0.0065	0.6897 ± 0.0057	0.1927 ± 0.0031	-0.5714 ± 0.0106	0.0781 ± 0.0903
	CoxTime	0.6941 ± 0.0052	0.1778 ± 0.0045	-0.5336 ± 0.0140	0.7508 ± 0.0092	0.1147 ± 0.0023	-0.3680 ± 0.0071	0.7061 ± 0.0052	0.1884 ± 0.0031	-0.5573 ± 0.0081	0.1508 ± 0.2010
	DeepHit	0.6847 ± 0.0063	0.1914 ± 0.0041	-0.5653 ± 0.0102	0.7082 ± 0.0083	0.1238 ± 0.0021	-0.3954 ± 0.0055	0.6875 ± 0.0070	0.2090 ± 0.0020	-0.6042 ± 0.0045	0.0657 ± 0.0990
	NnetSurv	0.6722 ± 0.0058	**0.1773 ± 0.0043**	**-0.5293 ± 0.0117**	0.7115 ± 0.0060	**0.1137 ± 0.0017**	**-0.3650 ± 0.0049**	0.6827 ± 0.0059	**0.1883 ± 0.0030**	-0.5565 ± 0.0081	0.3449 ± 0.2590
	MDN	0.6746 ± 0.0057	0.1886 ± 0.0041	-0.5564 ± 0.0106	0.7017 ± 0.0123	0.1370 ± 0.0023	-0.4407 ± 0.0060	0.6853 ± 0.0075	0.2138 ± 0.0022	-0.6146 ± 0.0055	0.0000 ± 0.0000
	DeSurv	**0.6950 ± 0.0048**	0.1792 ± 0.0053	-0.5357 ± 0.0157	0.7501 ± 0.0094	0.1163 ± 0.0032	-0.3723 ± 0.0092	0.7067 ± 0.0055	0.1897 ± 0.0035	-0.5597 ± 0.0083	0.0160 ± 0.0420
	QSurv	0.6936 ± 0.0063	0.1778 ± 0.0042	-0.5308 ± 0.0110	**0.7512 ± 0.0077**	0.1154 ± 0.0028	-0.3697 ± 0.0080	**0.7067 ± 0.0059**	0.1885 ± 0.0032	**-0.5562 ± 0.0075**	0.0583 ± 0.1195
Average Rank (Tabular)	CoxCC	4.33	3.00	3.08	4.33	3.42	3.67	4.08	3.17	3.50	
	CoxTime	3.00	3.58	3.17	3.00	3.33	2.58	2.75	3.17	3.00	
	DeepHit	5.17	6.50	6.33	5.83	5.50	5.50	5.50	6.17	6.00	
	NnetSurv	6.33	4.58	5.08	6.00	5.17	5.00	6.33	5.17	5.17	
	MDN	4.50	4.83	5.00	4.83	6.00	6.33	4.67	5.67	6.00	
	DeSurv	**2.17**	3.33	3.33	2.42	3.17	3.42	2.67	3.00	3.00	
	QSurv	2.50	**2.17**	**2.00**	**1.58**	**1.42**	**1.50**	**2.00**	**1.67**	**1.33**	

summarized in Appendix F. Model performance was evaluated using the time-dependent concordance index (C_{td}), Integrated Brier Score (IBS), Integrated Binomial Log-Likelihood (IBLL), and D-calibration. Higher C_{td} and IBLL and lower IBS indicate better performance. We define these metrics in Appendix F.3. To assess performance across follow-up durations, we evaluated each method over three horizons: the full horizon defined by $\hat{G}(\tau) = 0.001$, and two restricted quantile-based horizons, T_{Q_1} and T_{Q_2}, with T_{Q_2} corresponding to the median horizon. For robust comparison, we performed random-search hyperparameter optimization with 30 configurations per model for tabular datasets and 20 configurations per model for imaging datasets. The full procedure was repeated 20 times for tabular datasets and 5 times for imaging datasets. Due to the high computational burden of adaptive ODE integration under this evaluation protocol, SODEN was evaluated on the imaging datasets but omitted from the tabular experiments.

Table 2 shows that QSurv is consistently competitive across datasets, horizons, and evaluation metrics. Careful hyperparameter optimization makes several baselines strong, including CoxCC, CoxTime, NnetSurv, DeepHit, MDN, and DeSurv. Nevertheless, QSurv frequently achieves the best or second-

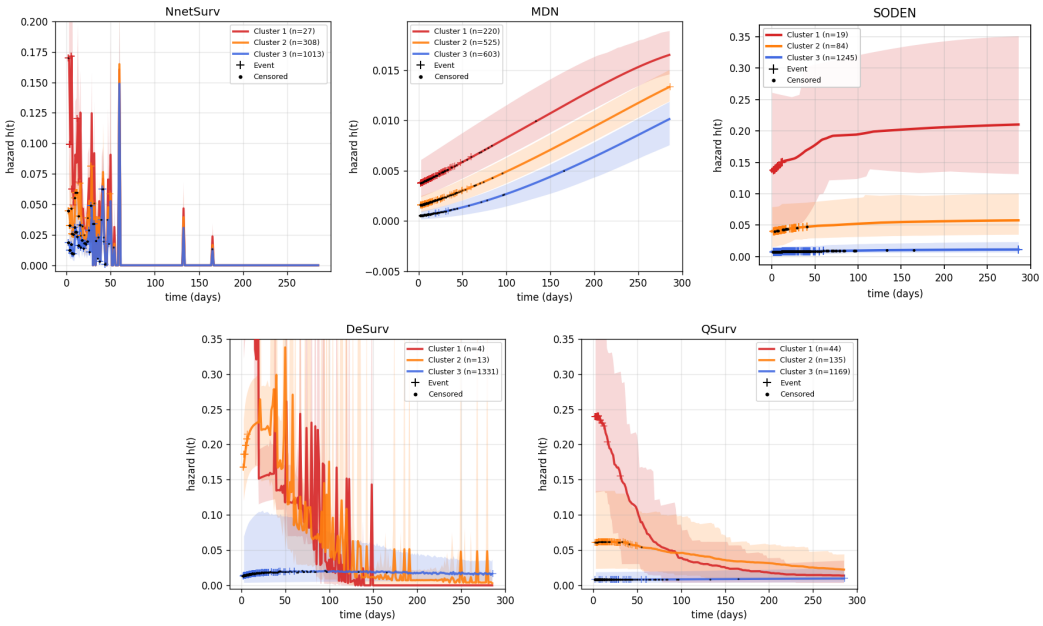

Figure 1: Predicted instantaneous hazard functions on COVID-19-NY across representative risk clusters. Lines show cluster-level predicted hazards, with event times marked by plus signs and censored observations by dots. NnetSurv produces irregular spike-like hazards, MDN produces smooth but constrained trajectories, SODEN produces slowly varying hazards, and DeSurv captures temporal variation but with pronounced oscillations and spikes. QSurv yields smoother and more interpretable cluster-specific temporal patterns, including elevated early risk in the high-risk cluster followed by gradual decline over time.

best performance, particularly for discrimination and survival probability estimation at restricted horizons. QSurv also has the best average rank on both the imaging and the tabular datasets. These results suggest that directly parameterizing the instantaneous hazard, while evaluating the cumulative hazard through quadrature, provides a robust balance between flexible time-to-event modeling and stable likelihood-based training.

The advantage of QSurv is especially clear in the high-dimensional imaging settings, where flexible time-varying hazard modeling is most relevant. On COVID-19-NY, QSurv achieves the best C_{td} at all three horizons, while remaining competitive with DeSurv and SODEN on likelihood- and calibration-oriented metrics. On C4KC-KiTS, QSurv obtains the best C_{td} across all horizons and the strongest short-horizon IBS and IBLL. On BraTS, QSurv also achieves the best C_{td} across all horizons and strong survival probability estimation, including the best median-horizon IBS and IBLL. These findings support the main motivation of QSurv: preserving continuous event times and directly learning instantaneous hazard can be beneficial when risk evolves nonlinearly over follow-up and covariates are high-dimensional.

On tabular datasets, performance differences are generally smaller, and several classical or discrete-time baselines remain highly competitive. QSurv nevertheless remains stable across datasets and horizons, often matching or improving on the best-performing alternatives. This suggests that the proposed quadrature-based hazard formulation does not sacrifice robustness in lower-dimensional settings, while providing additional flexibility for settings where proportional-hazards, parametric, or fixed-grid assumptions may be restrictive.

To further examine model behavior beyond aggregate prediction metrics, we compared predicted instantaneous hazard functions across representative risk clusters on COVID-19-NY (Figure 1). Hazard clusters were formed by applying three-cluster K-means to subject-specific predicted hazard trajectories evaluated over a common time grid. This analysis highlights qualitative differences in how each method characterizes time-resolved risk. NnetSurv produces highly irregular, spike-like hazards.

MDN yields smooth but relatively constrained, mostly monotone trajectories. SODEN produces smooth hazards that rise only slowly and separate clusters mainly by risk level. DeSurv captures temporal variation, but its hazards remain irregular, with sharp oscillations and isolated spikes over follow-up. In contrast, QSurv produces smoother and more interpretable cluster-specific patterns: a high-risk cluster with elevated early hazard followed by gradual decline, an intermediate-risk cluster with persistent moderate risk, and a low-risk cluster with near-flat hazard. Unlike survival curves, which summarize accumulated risk up to each time point, these hazard trajectories directly identify when risk is concentrated over follow-up. The early high-risk pattern estimated by QSurv aligns with COVID-19 studies reporting concentrated early excess mortality after infection and elevated readmission-or-death risk shortly after hospital discharge, with attenuation over time [28, 13].

This comparison suggests that the qualitative difference between QSurv and SODEN may arise partly from how the cumulative hazard is evaluated during training, rather than from the hazard parameterization alone. Compared with SODEN, the fixed Gauss-Legendre quadrature rule evaluates the hazard at predetermined points over each subject's observed interval, rather than relying on adaptive solver behavior. Compared with DeSurv, QSurv treats the hazard itself as the primitive modeled quantity rather than recovering it indirectly from a learned distribution function. The resulting hazard estimates are flexible enough to capture non-monotone and cluster-specific temporal risk patterns, while remaining substantially smoother and more stable than the discrete-time or indirectly derived alternatives. QSurv is most useful when the shape of the hazard over follow-up is of direct interest, as in the COVID-19-NY example above. In our experiments its gains were larger on the imaging datasets than on the tabular datasets, where several baselines performed comparably.

We further compared Time-LoRA with FiLM on all nine datasets, and with concatenation on the tabular datasets, under the same protocol as Table 2 (Table 6). On the tabular datasets FiLM and concatenation perform comparably to Time-LoRA, and every paired C_{td} difference has a 95% confidence interval that includes zero. On the imaging datasets Time-LoRA ranks ahead of FiLM on average, although five repetitions do not resolve the differences. The simulation ablation is reported in Appendix E.

5 Conclusion

We introduced QSurv, a quadrature-based nonparametric continuous-time survival model that directly parameterizes the instantaneous hazard as a flexible function of time and covariates. By evaluating the cumulative hazard with Gauss-Legendre quadrature, QSurv enables likelihood-based training without proportional-hazards assumptions, parametric survival distributions, time discretization, or adaptive ODE solvers. The resulting objective is simple, differentiable, and compatible with minibatch SGD, while Time-LoRA extends the framework to high-dimensional backbones by efficiently conditioning hazard predictions on time. Across simulations, tabular benchmarks, and imaging datasets, QSurv achieves competitive and robust performance. More importantly, its learned hazards provide time-resolved risk characterization, revealing clinically interpretable temporal patterns that may be obscured by aggregate survival summaries. A remaining limitation is that QSurv, like other deep survival models, depends on the hazard being learnable from available covariates and may require further validation under severe censoring, distribution shift, or highly irregular hazard dynamics. These results support QSurv as a practical framework for scalable and flexible continuous-time hazard modeling.

Acknowledgments and Disclosure of Funding

The research of Chaeyeon Lee and Sehwan Kim was supported by the Global-Learning & Academic Research Institution for Master's and PhD Students, and Postdocs (G-LAMP) Program of the National Research Foundation of Korea (NRF), funded by the Ministry of Education (No. RS-2025-25442252).

The computational requirements for this work were supported in part by the NYU Langone High Performance Computing (HPC) Core's resources.

References

[1] A. Avati, T. Duan, S. Zhou, K. Jung, N. H. Shah, and A. Y. Ng. Countdown regression: Sharp and calibrated survival predictions. In R. P. Adams and V. Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 145–155. PMLR, 22–25 Jul 2020.

[2] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Scientific data*, 4(1):170117, 2017.

[3] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. *arXiv preprint arXiv:1811.02629*, 2018.

[4] A. Bennis, S. Mouysset, and M. Serrurier. Estimation of conditional mixture weibull distribution with right censored data using neural network for time-to-event analysis. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I*, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-47425-6. doi: 10.1007/978-3-030-47426-3_53. URL `https://doi.org/10.1007/978-3-030-47426-3_53`.

[5] N. E. Breslow and N. Chatterjee. Design and analysis of two-phase studies with binary outcome applied to wilms tumour prognosis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48(4):457–468, 1999.

[6] A. F. Connors, N. V. Dawson, N. A. Desbiens, W. J. Fulkerson, L. Goldman, W. A. Knaus, J. Lynn, R. K. Oye, M. Bergner, A. Damiano, et al. A controlled trial to improve care for seriously iii hospitalized patients: The study to understand prognoses and preferences for outcomes and risks of treatments (support). *Jama*, 274(20):1591–1598, 1995.

[7] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972. ISSN 00359246. URL `http://www.jstor.org/stable/2985181`.

[8] E. Craig, C. Zhong, and R. Tibshirani. Survival stacking: casting survival analysis as a classification problem, 2021. URL `https://arxiv.org/abs/2107.13480`.

[9] C. Curtis, S. P. Shah, S.-F. Chin, G. Turashvili, O. M. Rueda, M. J. Dunning, D. Speed, A. G. Lynch, S. Samarajiwa, Y. Yuan, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, 486(7403):346–352, 2012.

[10] D. Danks and C. Yau. Derivative-based neural modelling of cumulative distribution functions for survival analysis. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 7240–7256. PMLR, 28–30 Mar 2022. URL `https://proceedings.mlr.press/v151/danks22a.html`.

[11] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

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

[13] J. P. Donnelly, X. Q. Wang, T. J. Iwashyna, and H. C. Prescott. Readmission and death after initial hospital discharge among patients with covid-19 in a large multihospital system. *Jama*, 325(3):304–306, 2021.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

[16] M. F. Gensheimer and B. Narasimhan. A scalable discrete-time survival model for neural networks. *PeerJ*, 7:e6257, 2019.

[17] G. H. Golub and J. H. Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

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

[19] H. Haider, B. Hoehn, S. Davis, and R. Greiner. Effective ways to build and evaluate individual survival distributions. *Journal of Machine Learning Research*, 21(85):1–63, 2020.

[20] X. Han, M. Goldstein, and R. Ranganath. Survival mixture density networks. In *Machine Learning for Healthcare Conference*, pages 224–248. PMLR, 2022.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] N. Heller, N. Sathianathen, A. Kalapara, E. Walczak, K. Moore, H. Kaluzniak, J. Rosenberg, P. Blake, Z. Rengel, M. Oestreich, J. Dean, M. Tradewell, A. Shah, R. Tejpaul, Z. Edgerton, M. Peterson, S. Raza, S. Regmi, N. Papanikolopoulos, and C. Weight. Data from C4KC-KiTS. The Cancer Imaging Archive, 2019. URL `https://doi.org/10.7937/TCIA.2019.IX49E8NX`.

[23] M. A. Hernán. The hazards of hazard ratios. *Epidemiology*, 21(1):13–15, 2010.

[24] K. R. Hess and V. A. Levin. Getting more out of survival data by using the hazard function. *Clinical Cancer Research*, 20(6):1404–1409, 2014.

[25] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[27] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841 – 860, 2008. doi: 10.1214/08-AOAS169. URL `https://doi.org/10.1214/08-AOAS169`.

[28] T. J. Iwashyna, S. Seelye, T. S. Berkowitz, J. Pura, A. S. Bohnert, C. B. Bowling, E. J. Boyko, D. M. Hynes, G. N. Ioannou, M. L. Maciejewski, et al. Late mortality after covid-19 infection among us veterans vs risk-matched comparators: a 2-year cohort analysis. *JAMA internal medicine*, 183(10):1111–1119, 2023.

[29] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958. ISSN 01621459. URL `http://www.jstor.org/stable/2281868`.

[30] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):24, 2018.

[31] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[32] H. Kvamme, Ø. Borgan, and I. Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of machine learning research*, 20(129):1–30, 2019.

[33] C. Lee, W. Zame, J. Yoon, and M. Van Der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[34] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015. doi: 10.1109/TMI.2014.2377694.

[35] C. Nagpal, X. Li, and A. 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.

[36] C. Nagpal, S. Yadlowsky, N. Rostamzadeh, and K. Heller. Deep cox mixtures for survival regression. In *Machine Learning for Healthcare Conference*, pages 674–708. PMLR, 2021.

[37] National Heart, Lung, and Blood Institute. About BioLINCC. Online, 2022. URL `https://biolincc.nhlbi.nih.gov/about/`.

[38] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[39] R. Ranganath, A. Perotte, N. Elhadad, and D. Blei. Deep survival analysis. In F. Doshi-Velez, J. Fackler, D. Kale, B. Wallace, and J. Wiens, editors, *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 101–114, Northeastern University, Boston, MA, USA, 18–19 Aug 2016. PMLR.

[40] P. Royston. Flexible parametric alternatives to the cox model, and more. *The Stata Journal*, 1 (1):1–28, 2001.

[41] J. Saltz, M. Saltz, P. Prasanna, R. Moffitt, J. Hajagos, E. Bremer, J. Balsamo, and T. Kurc. Stony Brook University COVID-19 Positive Cases. The Cancer Imaging Archive, 2021. URL `https://doi.org/10.7937/TCIA.BBAG-2923`.

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

[43] M. J. Stensrud and M. A. Hernán. Why test for proportional hazards? *Jama*, 323(14):1401–1402, 2020.

[44] W. Tang, J. Ma, Q. Mei, and J. Zhu. Soden: A scalable continuous-time survival model through ordinary differential equation networks. *Journal of Machine Learning Research*, 23(34):1–29, 2022.

[45] W. Tang, K. He, G. Xu, and J. Zhu. Survival analysis via ordinary differential equations. *Journal of the American Statistical Association*, 118(544):2406–2421, 2023.

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

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[48] S. Wiegrebe, P. Kopper, R. Sonabend, B. Bischl, and A. Bender. Deep learning for survival analysis: a review. *Artificial Intelligence Review*, 57(3):65, 2024.

[49] C.-N. Yu, R. Greiner, H.-C. Lin, and V. Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. *Advances in neural information processing systems*, 24, 2011.

[50] Q. Zhong, J. W. Mueller, and J.-L. Wang. Deep extended hazard models for survival analysis. *Advances in Neural Information Processing Systems*, 34:15111–15124, 2021.

A Gauss-Legendre Quadrature

Gauss-Legendre quadrature approximates a definite integral by a weighted sum of function evaluations at carefully chosen nonuniform nodes. Unlike grid-based rules such as Riemann sums or the trapezoidal rule, which evaluate the integrand at equally spaced points, Gauss-Legendre quadrature chooses nodes and weights to maximize algebraic precision. For an integral over the standard interval $[-1, 1]$, the K-point rule is

$$\int_{-1}^{1} q(u)du \approx \sum_{k=1}^{K} w_k q(\xi_k), \tag{9}$$

where ξ_k are the roots of the K-th Legendre polynomial P_K and

$$w_k = \frac{2}{(1-\xi_k^2)[P_K'(\xi_k)]^2}. \tag{10}$$

The nodes and weights are fixed by the quadrature rule, not learned from data. With K function evaluations, the rule is exact for all polynomials of degree at most $2K-1$.

For an integral over a general interval $[a, b]$, the affine transformation

$$s = \frac{b-a}{2}u + \frac{b+a}{2} \tag{11}$$

maps $u \in [-1, 1]$ to $s \in [a, b]$. Therefore,

$$\int_a^b q(s)ds \approx \frac{b-a}{2} \sum_{k=1}^{K} w_k q\left(\frac{b-a}{2}\xi_k + \frac{b+a}{2}\right). \tag{12}$$

When q is $2K$ times continuously differentiable, the classical Gauss-Legendre error formula gives

$$E_K(q) = \frac{(b-a)^{2K+1}(K!)^4}{(2K+1)[(2K)!]^3} q^{(2K)}(\zeta) \tag{13}$$

for some $\zeta \in (a, b)$. Thus, the approximation error is controlled by the high-order smoothness of the integrand. For analytic functions, Gauss-Legendre quadrature can converge much faster than low-order grid-based rules, often requiring substantially fewer evaluations to reach a comparable accuracy.

In QSurv, the integrand is the neural network parameterized hazard $q(s) = \lambda_\theta(s|x)$ and the interval is subject-specific, $[0, o_i]$. Applying the above transformation with $a = 0$ and $b = o_i$ gives

$$\Lambda_\theta(o_i|x_i) = \int_0^{o_i} \lambda_\theta(s|x_i)ds \approx \frac{o_i}{2} \sum_{k=1}^{K} w_k \lambda_\theta(o_i\tau_k|x_i), \tag{14}$$

where $\tau_k = (\xi_k + 1)/2$. This is the approximation used in the QSurv likelihood: the hazard remains a continuous-time function, while the cumulative hazard is evaluated through a fixed number of differentiable neural network parameterized hazard evaluations.

B Proofs

Proof of Theorem 3.1. The cumulative hazard is defined as the integral $\Lambda_\theta(t|x) = \int_0^t \lambda_\theta(s|x)\, ds$. To apply the standard error bound for Gauss-Legendre quadrature, which is defined on the reference interval $[-1, 1]$, we perform an affine transformation of variables. Let $s(u) = \frac{t}{2}(u+1)$. This maps $u \in [-1, 1]$ to $s \in [0, t]$. The differential is $ds = \frac{t}{2}\, du$. The standard error term for K-point Gauss-Legendre quadrature on $[-1, 1]$ for a function g is given by $E_K(g) = \frac{2^{2K+1}(K!)^4}{(2K+1)[(2K)!]^3}\, g^{(2K)}(\xi)$ for some $\xi \in (-1, 1)$. Let $g(u) = \lambda_\theta(\frac{t}{2}(u+1)|x)$. By the chain rule, the $2K$-th derivative scales as $g^{(2K)}(u) = (\frac{t}{2})^{2K}\lambda_\theta^{(2K)}(s(u)|x)$. Evaluating at the point $\xi \in (-1, 1)$ given by the quadrature error theorem and letting $\tau = s(\xi) = (t/2)(\xi+1) \in (0, t)$, we obtain $g^{(2K)}(\xi) = (t/2)^{2K}\, \lambda_\theta^{(2K)}(\tau|x)$. The total error is the standard error scaled by the integration measure $\frac{t}{2}$:

$$\text{Error} = \frac{t}{2} \cdot \frac{2^{2K+1}(K!)^4}{(2K+1)[(2K)!]^3} \cdot \left[\left(\frac{t}{2}\right)^{2K} \lambda_\theta^{(2K)}(\tau|x)\right]. \tag{15}$$

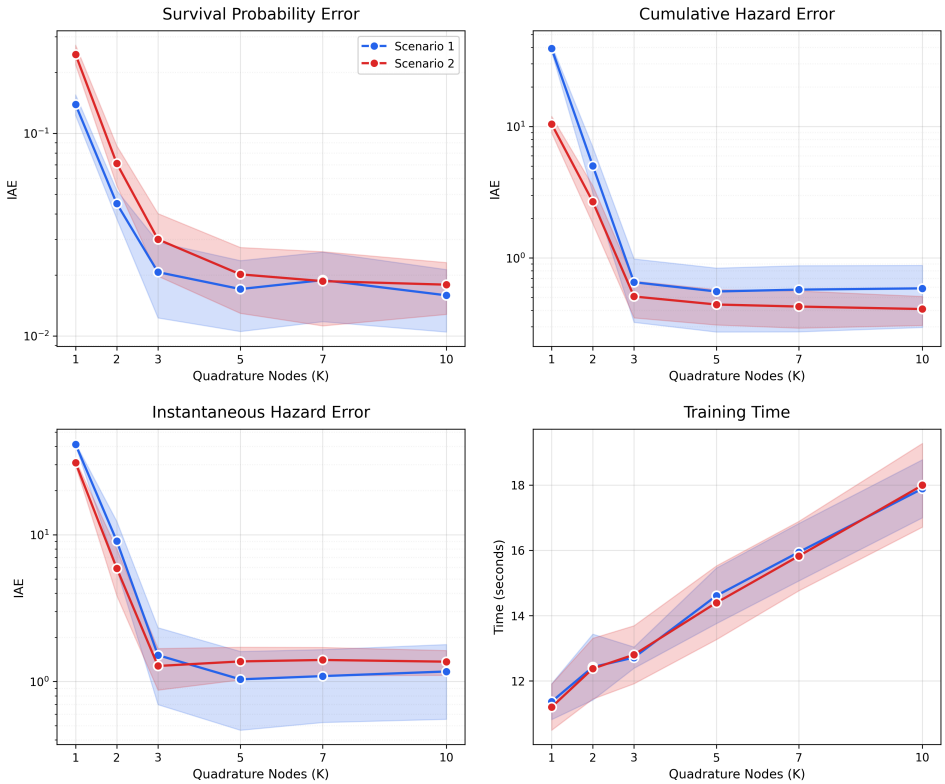

Figure 2: Convergence of approximation error and training efficiency on two simulation scenarios. The first three plots display the integrated absolute error (IAE) for survival probability, cumulative hazard, and instantaneous hazard as a function of quadrature nodes (K). All metrics exhibit rapid convergence, saturating beyond $K = 5$ across both simulation scenarios. The fourth panel shows that training time increases linearly with K. Shaded regions denote standard deviation ($n = 20$).

Grouping terms, the factors of 2 cancel, leaving the coefficient $\frac{t^{2K+1}(K!)^4}{(2K+1)[(2K)!]^3}$. Bounding $|\lambda_\theta^{(2K)}(\tau|x)|$ by $\max_{\tau\in[0,t]} |\lambda_\theta^{(2K)}(\tau|x)|$ yields the inequality. □

Proof of Corollary 3.2. The negative log-likelihood decomposes into a log-hazard term and a cumulative hazard term. Since the hazard function $\lambda_\theta(t|x)$ is computed exactly via the neural network forward pass at time t, the term $-\delta \log \lambda_\theta(t|x)$ is identical in both the true and approximate losses. Consequently, the difference depends solely on the integral approximation:

$$|\mathcal{L} - \hat{\mathcal{L}}_K| = |(-\delta \log \lambda + \Lambda) - (-\delta \log \lambda + \hat{\Lambda}_K)| = |\Lambda - \hat{\Lambda}_K| = \varepsilon_K(t|x).$$

□

C Number of Quadrature Nodes

We further investigate how the number of quadrature nodes K affects the estimation of the instantaneous hazard, cumulative hazard, and survival function. Although QSurv represents $\lambda_\theta(t|x)$ as a continuous-time neural function, the training objective evaluates this function only at finitely many time points: the observed event times through the log-hazard contribution and the subject-specific quadrature nodes through the cumulative-hazard contribution. Therefore, if K is too small relative to the temporal complexity of the hazard, the quadrature-based objective may fail to resolve rapid changes in risk over time.

This observation clarifies the role of K in QSurv. Unlike the number of bins in discrete-time survival models, K does not define the temporal resolution of the hazard model itself. The hazard remains a

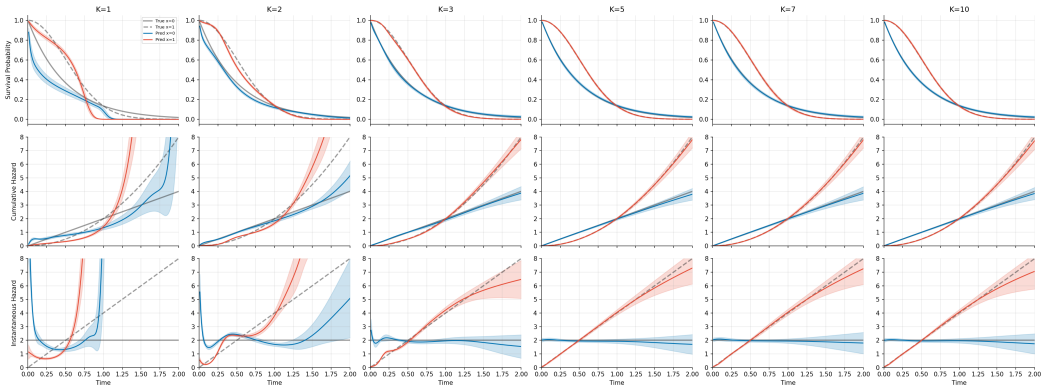

Figure 3: Instantaneous hazard, cumulative hazard, and survival functions for simulation scenario 1.

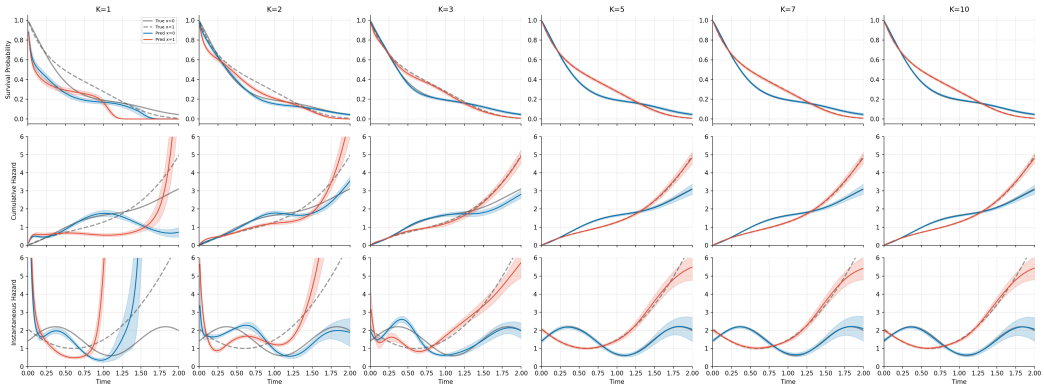

Figure 4: Instantaneous hazard, cumulative hazard, and survival functions for simulation scenario 2.

continuous function of time. Instead, K controls the numerical precision with which the cumulative hazard is approximated inside the likelihood. Reducing K should therefore not be interpreted as imposing a simpler or more regularized hazard model. Rather, very small values of K may lead to a poor approximation of the continuous-time likelihood, especially when the hazard has sharp or non-monotone temporal structure. Conversely, once K is large enough to resolve the relevant temporal variation, increasing K further should have little effect on estimation accuracy, aside from increased computation.

To empirically assess this behavior, we conducted a simulation study using two synthetic scenarios with complex, non-monotone hazards. Scenario 1 is a crossing-hazards setting with a binary covariate $x \sim \text{Bernoulli}(0.5)$, defined by

$$S(t|x) = \exp(-t^{1+x}), \qquad \lambda(t|x) = (1+x)t^x. \tag{16}$$

Thus, the $x = 0$ group follows $S(t|0) = \exp(-t)$ with constant hazard $\lambda(t|0) = 1$, whereas the $x = 1$ group follows $S(t|1) = \exp(-t^2)$ with linearly increasing hazard $\lambda(t|1) = 2t$. The hazards intersect at $t = 0.5$, while the survival curves cross at $t = 1$, where $S(t|0) = S(t|1) = e^{-1}$. Censoring times are sampled from $C \sim \text{Uniform}(0, 2)$.

Scenario 2 introduces higher-frequency temporal variation through anti-phase sinusoidal hazards:

$$\begin{aligned} \lambda(t|x) &= 1 + 0.8(1-2x)\sin(4t), \\ \Lambda(t|x) &= t + 0.2(1-2x)\{1-\cos(4t)\}, \\ S(t|x) &= \exp(-\Lambda(t|x)). \end{aligned} \tag{17}$$

For $x = 0$, the hazard is $1 + 0.8\sin(4t)$, while for $x = 1$, the hazard is $1 - 0.8\sin(4t)$. This scenario creates oscillating risk dynamics with multiple hazard intersections, requiring the model to resolve rapid temporal changes in risk. Survival times are generated by inverse transform sampling, with censoring drawn from an exponential distribution with rate approximately 0.33.

Table 3: Sensitivity of QSurv to the number of Gauss–Legendre nodes K on the tabular datasets, with each K trained and tuned under the same protocol as Table 2. Values are mean $\pm$ SD over seeds.

Dataset	K=2	K=3	K=5	K=10	K=15
$C_{td} \uparrow$					
FLCHAIN	0.7955 ± 0.0103	0.7962 ± 0.0103	0.7965 ± 0.0100	0.7962 ± 0.0097	0.7963 ± 0.0097
FRAMINGHAM	0.7419 ± 0.0141	0.7404 ± 0.0142	0.7396 ± 0.0147	0.7392 ± 0.0149	0.7398 ± 0.0137
GBSG	0.6563 ± 0.0263	0.6553 ± 0.0245	0.6552 ± 0.0246	0.6560 ± 0.0242	0.6554 ± 0.0264
METABRIC	0.6591 ± 0.0187	0.6562 ± 0.0188	0.6599 ± 0.0204	0.6558 ± 0.0193	0.6591 ± 0.0188
NWTCO	0.6364 ± 0.0280	0.6431 ± 0.0236	0.6433 ± 0.0249	0.6440 ± 0.0263	0.6467 ± 0.0297
SUPPORT2	0.6735 ± 0.0091	0.6809 ± 0.0064	0.6908 ± 0.0054	0.6924 ± 0.0055	0.6928 ± 0.0063
IBS $\downarrow$					
FLCHAIN	0.0936 ± 0.0026	0.0935 ± 0.0026	0.0935 ± 0.0026	0.0936 ± 0.0024	0.0935 ± 0.0025
FRAMINGHAM	0.0881 ± 0.0028	0.0886 ± 0.0028	0.0887 ± 0.0028	0.0889 ± 0.0030	0.0886 ± 0.0027
GBSG	0.1779 ± 0.0155	0.1767 ± 0.0147	0.1778 ± 0.0141	0.1768 ± 0.0123	0.1763 ± 0.0128
METABRIC	0.1553 ± 0.0096	0.1553 ± 0.0117	0.1555 ± 0.0125	0.1561 ± 0.0112	0.1553 ± 0.0110
NWTCO	0.1147 ± 0.0040	0.1118 ± 0.0028	0.1122 ± 0.0027	0.1119 ± 0.0029	0.1117 ± 0.0028
SUPPORT2	0.1902 ± 0.0061	0.1854 ± 0.0076	0.1801 ± 0.0043	0.1784 ± 0.0048	0.1785 ± 0.0056

We trained QSurv models with $K \in \{1, 2, 3, 5, 7, 10\}$ quadrature nodes across 20 independent runs. The results are summarized in Figure 2. We observe a rapid decrease in integrated absolute error for the instantaneous hazard, cumulative hazard, and survival function as K increases. The error curves stabilize after a moderate number of quadrature nodes, indicating that the quadrature approximation has become sufficiently accurate for these settings. As expected, training time increases approximately linearly with K, reflecting the additional neural network parameterized hazard evaluations required at the quadrature nodes.

Figures 3 and 4 show the corresponding estimated trajectories. With very small values of K, the fitted hazard can exhibit visible approximation artifacts and may fail to capture important temporal features of the true hazard. In contrast, moderate values of K recover stable estimates of the instantaneous hazard, cumulative hazard, and survival function. These findings support our practical use of a fixed moderate quadrature order in the main experiments.

Table 3 reports C_{td} and IBS on the tabular datasets for $K \in \{2, 3, 5, 10, 15\}$. Predictive performance changes little across these values of K.

Table 4: Data generation mechanisms. All covariate-dependent parameters are modeled as $\theta(x) = \exp(\mathcal{P}(x; \mathbf{w}))$ to ensure positivity, except for $\mu(x)$ in the Log-Normal distribution, which is modeled directly as $\mu(x) = \mathcal{P}(x; \mathbf{w})$. The polynomial function is defined as $\mathcal{P}(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$.

Distribution	**Hazard / PDF Definition**	**Parameter Coefficients**
Exponential	$h(t\|x) = \lambda(x)$	$\mathbf{w}_\lambda = [-1.0, 0.5, -0.3, 0.15]$
Weibull	$h(t\|x) = \frac{k(x)}{\lambda(x)} \left(\frac{t}{\lambda(x)}\right)^{k(x)-1}$	$\mathbf{w}_k = [0.3, 0.2, -0.1, 0.05]$ $\mathbf{w}_\lambda = [2.0, 0.3, -0.2, 0.1]$
Gamma	$f(t\|x) = \frac{\beta(x)^{k(x)}}{\Gamma(k(x))} t^{k(x)-1} e^{-\beta(x)t}$	$\mathbf{w}_k = [1.8, 0.3, -0.1, 0.05]$ $\mathbf{w}_\beta = [0.3, -0.4, 0.15, -0.05]$
Gompertz	$h(t\|x) = b(x)e^{ct}$	$\mathbf{w}_b = [-2.0, 0.4, -0.2, 0.1]$ $c = 0.05$
Log-Normal	$T = \exp(\mu(x) + \sigma(x)Z),\ Z \sim \mathcal{N}(0, 1)$	$\mathbf{w}_\mu = [1.5, 0.8, -0.4, 0.2]$ $\mathbf{w}_\sigma = [-0.1, 0.25, -0.10, 0.03]$
Log-Logistic	$h(t\|x) = \frac{(\beta(x)/\alpha(x))(t/\alpha(x))^{\beta(x)-1}}{1+(t/\alpha(x))^{\beta(x)}}$	$\mathbf{w}_\alpha = [1.2, 0.4, -0.15, 0.08]$ $\mathbf{w}_\beta = [1.0, 0.3, -0.1, 0.05]$

D Simulation Study

To evaluate the capability of the proposed models to capture complex, non-linear relationships between covariates and survival outcomes, we generated synthetic datasets using six parametric distributions: Exponential, Weibull, Gamma, Gompertz, Log-Normal, and Log-Logistic. For all scenarios, we simulated a one-dimensional covariate x drawn from a uniform distribution between -1 and 1. The distribution parameters, such as shape, scale, or rate, were modeled as non-linear functions of x. We employed third-order polynomial transformations followed by an exponential link function to ensure positivity, except for the location parameter in the Log-Normal distribution. The polynomial coefficients were manually tuned to ensure diverse hazard shapes and stable event times within the observation window.

The chosen distributions cover a wide range of survival behaviors. The Exponential and Gompertz scenarios represent Proportional Hazards settings, whereas the Weibull, Gamma, Log-Normal, and Log-Logistic distributions induce non-proportional hazards where the shape of the distribution changes conditional on the covariate. Notably, the Log-Normal and Log-Logistic distributions generate unimodal hazard functions that rise and then fall, mimicking patterns often seen in medical contexts.

Censoring times were generated independently of the covariates using a uniform distribution, where the upper bound was dynamically calibrated for each dataset to achieve a target censoring rate of approximately 20%. The observed time was defined as the minimum of the true event time and the censoring time. For each simulation run, we generated 2,000 training and testing samples, respectively. The experiments were repeated 20 times with different random seeds to report the mean and standard deviation of the results. All neural network baselines used a standardized architecture with two hidden layers of 32 units and Tanh activations. To strictly assess how well the models recovered the underlying distributions, we calculated the L1 error for the hazard, cumulative hazard, and survival functions on the testing samples by comparing the model estimates against the true analytical functions.

To assess the model's ability to recover covariate-averaged survival dynamics, we averaged the conditional survival, cumulative hazard, and instantaneous hazard functions over the covariate distribution $p(x)$. The population averages and their empirical model-based approximations are:

$$\begin{aligned}
\bar{S}(t) &= \int_{\mathcal{X}} S(t|x)p(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} \hat{S}(t|x_i), \\
\bar{\Lambda}(t) &= \int_{\mathcal{X}} \Lambda(t|x)p(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} \hat{\Lambda}(t|x_i), \\
\bar{\lambda}(t) &= \int_{\mathcal{X}} \lambda(t|x)p(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} \hat{\lambda}(t|x_i).
\end{aligned} \tag{18}$$

Here, $\bar{S}(t)$ is the marginal survival function, while $\bar{\Lambda}(t)$ and $\bar{\lambda}(t)$ are covariate-averaged conditional cumulative hazard and hazard functions. The latter two generally differ from the cumulative hazard and hazard associated with $\bar{S}(t)$.

By comparing the model-based averages with the corresponding averages of the true analytical functions, we assess how well the model recovers the average conditional hazard shape and population survival decay when the conditional distributions depend nonlinearly on the covariates. Figures 6–11 present the results.

E LoRA Ablation Study

We performed an ablation study to compare different time-conditioning mechanisms within QSurv. Specifically, we compared three variants: direct concatenation of time and covariates, FiLM-based conditioning, and the proposed Time-LoRA conditioning. All variants used the same simulation settings, training protocol, and evaluation metric. Performance was measured by the L_1 error between the predicted and true instantaneous hazard functions, averaged over 20 random seeds.

Table 7 shows that no single conditioning mechanism uniformly dominates across all data-generating distributions. Direct concatenation performs best for Weibull, Gamma, and log-normal hazards,

Table 5: Performance comparison of survival models across various distributions. Values are Mean ± Std (20 seeds). Best results are **bold**, second best are underlined. Training times are wall-clock seconds on a single Apple M4 CPU (10-core, 32 GB RAM); relative comparisons across rows are meaningful but absolute values are machine-dependent.

Metric	Distribution	CoxCC	CoxTime	NnetSurv	MDN	DeSurv	SODEN	QSurv
L_1 Error of $\hat{S}$	Exponential	**0.0122 ± 0.0038**	0.0130 ± 0.0042	0.0266 ± 0.0084	0.0155 ± 0.0055	0.0177 ± 0.0057	0.0167 ± 0.0069	0.0162 ± 0.0058
	Weibull	0.0268 ± 0.0041	0.0144 ± 0.0038	0.0236 ± 0.0041	0.0354 ± 0.0037	**0.0127 ± 0.0043**	0.0139 ± 0.0053	0.0133 ± 0.0046
	Gamma	0.0197 ± 0.0041	0.0138 ± 0.0035	0.0213 ± 0.0032	0.1056 ± 0.0028	0.0121 ± 0.0035	**0.0107 ± 0.0029**	0.0110 ± 0.0030
	Gompertz	0.0121 ± 0.0040	0.0143 ± 0.0054	0.0276 ± 0.0058	0.0202 ± 0.0043	0.0129 ± 0.0043	**0.0115 ± 0.0034**	0.0120 ± 0.0032
	Log-Normal	0.0166 ± 0.0065	0.0166 ± 0.0049	0.0199 ± 0.0054	0.0272 ± 0.0034	**0.0152 ± 0.0054**	0.0165 ± 0.0054	0.0160 ± 0.0064
	Log-Logistic	0.0356 ± 0.0040	0.0157 ± 0.0042	0.0209 ± 0.0035	0.0567 ± 0.0041	**0.0144 ± 0.0041**	0.0150 ± 0.0042	0.0154 ± 0.0042
L_1 Error of $\hat{\Lambda}$	Exponential	**0.1780 ± 0.0479**	0.2013 ± 0.0638	0.4163 ± 0.1258	0.2244 ± 0.0958	0.2187 ± 0.0900	0.2144 ± 0.0827	0.2221 ± 0.0923
	Weibull	0.7486 ± 0.1803	0.2372 ± 0.1229	0.5065 ± 0.1114	0.5137 ± 0.1197	0.1974 ± 0.0644	**0.1887 ± 0.0842**	0.1894 ± 0.0938
	Gamma	2.4663 ± 0.3089	1.1946 ± 0.6717	1.3610 ± 0.4712	2.8405 ± 0.4260	0.6507 ± 0.2066	0.5796 ± 0.2705	**0.5766 ± 0.2472**
	Gompertz	0.2773 ± 0.1030	0.3699 ± 0.2559	0.7631 ± 0.1900	0.6015 ± 0.1340	0.2870 ± 0.1289	**0.1964 ± 0.1009**	0.1990 ± 0.1021
	Log-Normal	0.8253 ± 0.2925	0.6984 ± 0.3496	0.4052 ± 0.1085	1.0280 ± 0.4111	**0.3427 ± 0.1220**	0.4213 ± 0.1943	0.4205 ± 0.2091
	Log-Logistic	0.6392 ± 0.1234	0.2488 ± 0.0619	0.3262 ± 0.0438	0.4433 ± 0.0416	**0.2279 ± 0.1152**	0.2358 ± 0.1280	0.2305 ± 0.1158
L_1 Error of $\hat{\lambda}$	Exponential	0.1775 ± 0.0410	0.1775 ± 0.0422	0.3830 ± 0.0438	0.0568 ± 0.0446	0.0448 ± 0.0159	**0.0330 ± 0.0138**	0.0352 ± 0.0164
	Weibull	0.1576 ± 0.0322	0.1182 ± 0.0207	0.2383 ± 0.0254	0.0644 ± 0.0102	0.0295 ± 0.0090	**0.0272 ± 0.0119**	0.0282 ± 0.0139
	Gamma	0.9012 ± 0.0921	0.5661 ± 0.1769	0.7249 ± 0.0370	0.5646 ± 0.0459	0.2930 ± 0.0907	**0.1716 ± 0.0901**	0.1718 ± 0.0859
	Gompertz	0.1713 ± 0.0262	0.1890 ± 0.0549	0.2921 ± 0.0276	0.0920 ± 0.0140	0.0502 ± 0.0278	0.0262 ± 0.0174	**0.0258 ± 0.0180**
	Log-Normal	0.1235 ± 0.0248	0.1114 ± 0.0294	0.1938 ± 0.0219	0.1882 ± 0.0520	0.0473 ± 0.0201	0.0462 ± 0.0183	**0.0457 ± 0.0185**
	Log-Logistic	0.1621 ± 0.0210	0.1266 ± 0.0242	0.3290 ± 0.0525	0.1031 ± 0.0090	0.0513 ± 0.0299	0.0517 ± 0.0295	**0.0491 ± 0.0291**
Training Time (s)	Exponential	5.3210 ± 0.7844	6.7244 ± 0.8221	2.4345 ± 0.2233	3.5458 ± 0.2663	6.1273 ± 1.2448	87.9615 ± 8.5607	6.0749 ± 0.8276
	Weibull	5.0815 ± 0.4088	6.6068 ± 0.5126	2.4946 ± 0.4445	3.6100 ± 0.5023	5.9509 ± 0.6691	124.1380 ± 17.2815	5.7566 ± 0.4297
	Gamma	4.6292 ± 0.3867	6.3100 ± 0.6772	2.2909 ± 0.2931	3.4942 ± 0.4543	6.0968 ± 0.9452	149.9774 ± 24.4947	6.2316 ± 1.1584
	Gompertz	4.4591 ± 0.2230	5.9313 ± 0.3321	1.9610 ± 0.1278	3.0111 ± 0.0375	5.5127 ± 0.4924	82.9457 ± 10.5067	5.3545 ± 0.3065
	Log-Normal	5.8264 ± 2.2276	7.8568 ± 3.4891	2.8560 ± 1.1757	4.3230 ± 1.8299	8.7879 ± 6.4600	243.5298 ± 117.4764	8.0038 ± 4.6329
	Log-Logistic	4.3939 ± 0.8087	5.8667 ± 1.2868	2.3401 ± 1.8595	3.3814 ± 0.9135	5.0744 ± 0.6543	131.3422 ± 13.7360	5.7363 ± 0.6092

Table 6: Comparison of time-conditioning mechanisms in QSurv on the real-world datasets at the full horizon, under the same protocol as Table 2. Concatenation is reported for the tabular datasets only, and best results are **bold**.

Dataset	Model	C_{td} ↑	IBS ↓	IBLL ↑
COVID-19-NY	QSurv (FiLM)	0.6936 ± 0.1142	0.0948 ± 0.0414	-0.3029 ± 0.1168
	QSurv (LoRA)	**0.7456 ± 0.1046**	**0.0716 ± 0.0155**	**-0.2318 ± 0.0489**
C4KC-KiTS	QSurv (FiLM)	0.5995 ± 0.1437	**0.1111 ± 0.0133**	-0.4640 ± 0.1194
	QSurv (LoRA)	**0.6381 ± 0.1446**	0.1260 ± 0.0453	**-0.4002 ± 0.1113**
BraTS	QSurv (FiLM)	0.5364 ± 0.0323	0.1067 ± 0.0205	-0.3537 ± 0.0989
	QSurv (LoRA)	**0.5811 ± 0.0387**	**0.1038 ± 0.0133**	**-0.3327 ± 0.0371**
FLCHAIN	QSurv (Concat)	**0.7973 ± 0.0098**	**0.0934 ± 0.0024**	**-0.3099 ± 0.0072**
	QSurv (FiLM)	0.7967 ± 0.0101	0.0936 ± 0.0022	-0.3103 ± 0.0066
	QSurv (LoRA)	0.7965 ± 0.0099	0.0936 ± 0.0022	-0.3102 ± 0.0070
FRAMINGHAM	QSurv (Concat)	0.7412 ± 0.0161	0.0887 ± 0.0025	-0.2967 ± 0.0101
	QSurv (FiLM)	0.7414 ± 0.0156	**0.0882 ± 0.0033**	**-0.2933 ± 0.0100**
	QSurv (LoRA)	**0.7420 ± 0.0136**	0.0884 ± 0.0028	-0.2937 ± 0.0084
GBSG	QSurv (Concat)	0.6592 ± 0.0306	0.1935 ± 0.0804	-0.6272 ± 0.4759
	QSurv (FiLM)	0.6564 ± 0.0332	0.1773 ± 0.0134	-0.5248 ± 0.0358
	QSurv (LoRA)	**0.6597 ± 0.0273**	**0.1760 ± 0.0126**	**-0.5218 ± 0.0342**
METABRIC	QSurv (Concat)	0.6490 ± 0.0191	**0.1531 ± 0.0102**	**-0.4618 ± 0.0312**
	QSurv (FiLM)	**0.6586 ± 0.0206**	0.1543 ± 0.0097	-0.4620 ± 0.0274
	QSurv (LoRA)	0.6551 ± 0.0217	0.1558 ± 0.0122	-0.4664 ± 0.0337
NWTCO	QSurv (Concat)	0.6483 ± 0.0296	**0.1116 ± 0.0047**	**-0.3651 ± 0.0159**
	QSurv (FiLM)	0.6478 ± 0.0274	0.1120 ± 0.0031	-0.3663 ± 0.0117
	QSurv (LoRA)	**0.6495 ± 0.0271**	0.1118 ± 0.0032	-0.3664 ± 0.0127
SUPPORT2	QSurv (Concat)	0.6929 ± 0.0055	**0.1763 ± 0.0037**	-0.5306 ± 0.0117
	QSurv (FiLM)	0.6936 ± 0.0050	0.1775 ± 0.0040	**-0.5304 ± 0.0109**
	QSurv (LoRA)	**0.6937 ± 0.0055**	0.1776 ± 0.0039	-0.5310 ± 0.0107

suggesting that simple conditioning can be sufficient when the hazard structure is relatively easy to represent in the low-dimensional simulation setting. FiLM performs best for the exponential setting and remains competitive for Gompertz and log-logistic hazards. Time-LoRA achieves the

Table 7: Comparison of QSurv conditioning variants on L_1 error of instantaneous hazard (lower the better) across different parametric distributions, with standard deviations over 20 random seeds. Best results are **bold**, second best are underlined.

Distribution	QSurv (Concat)	QSurv (FiLM)	QSurv (LoRA)
Exponential	0.0327 ± 0.0159	$\mathbf{0.0260 \pm 0.0104}$	0.0352 ± 0.0164
Weibull	$\mathbf{0.0278 \pm 0.0055}$	0.0345 ± 0.0182	0.0282 ± 0.0139
Gamma	$\mathbf{0.1199 \pm 0.0344}$	0.2033 ± 0.0894	0.1718 ± 0.0859
Gompertz	0.0506 ± 0.0157	0.0421 ± 0.0403	$\mathbf{0.0258 \pm 0.0180}$
Log-normal	$\mathbf{0.0431 \pm 0.0154}$	0.0469 ± 0.0197	0.0457 ± 0.0185
Log-logistic	0.0939 ± 0.0283	0.0590 ± 0.0507	$\mathbf{0.0491 \pm 0.0291}$

best performance for Gompertz and log-logistic hazards, and is second best for Weibull, Gamma, and log-normal distributions.

These results suggest that Time-LoRA is a competitive time-conditioning mechanism for hazard modeling, particularly in settings where temporal risk patterns require more flexible interactions between time and covariates. At the same time, the ablation indicates that in simple low-dimensional simulations, direct concatenation can remain a strong baseline. The main motivation for Time-LoRA is therefore not that it uniformly improves all synthetic settings, but that it provides an efficient and flexible mechanism for conditioning high-dimensional neural backbones on time while preserving competitive hazard-estimation accuracy.

Figures 6–11 present the visualizations of the fitted curves.

F Experiment Details

F.1 Datasets

We evaluated all methods on a mixture of tabular and imaging survival datasets. The main benchmark includes six tabular datasets (GBSG, METABRIC, NWTCO, Framingham, FLCHAIN, and SUPPORT2) and three imaging datasets (COVID-19-NY, C4KC-KiTS, and BraTS).

GBSG comes from a multicenter randomized clinical trial conducted by the German Breast Cancer Study Group to evaluate adjuvant treatment strategies in patients with lymph node-positive breast cancer [42]. The endpoint is breast cancer recurrence after treatment. The public dataset contains 686 patients, with 299 observed recurrence events (43.6%) and 387 right-censored observations (56.4%). We used eight covariates covering demographic features, tumor characteristics, hormone receptor status, and treatment information. Categorical variables were one-hot encoded.

METABRIC comes from the Molecular Taxonomy of Breast Cancer International Consortium and was designed to study molecular subtypes of breast cancer [9]. We used the version distributed with Pycox, following the Immunohistochemical 4 plus Clinical (IHC4+C) covariate set [30]. The input variables include four gene-expression markers (MKI67, EGFR, PGR, and ERBB2) and five clinical variables (hormone therapy, radiotherapy, chemotherapy, estrogen receptor status, and age). The dataset contains 1,904 patients, with 1,103 deaths (57.9%) and 801 censored observations (42.1%). Missing categorical values were imputed with the most frequent category before one-hot encoding.

NWTCO is from the third and fourth National Wilms Tumor Study Group trials and includes pediatric patients diagnosed with Wilms tumor between 1979 and 1994 [5]. We analyzed 4,028 patients with complete relapse outcome and covariate information. The covariates include disease stage, age, and subcohort membership. There were 571 observed events (14.1%) and 3,457 censored observations (85.8%).

Framingham uses the Framingham Heart Study longitudinal teaching dataset distributed by NHLBI/BioLINCC [37]. This is a Framingham-derived teaching dataset rather than the full Framingham cohort. It contains longitudinal clinic, questionnaire, laboratory, and adjudicated event data on 4,434 participants from three examination periods, yielding 11,627 person-exam observations and 39 variables. Outcomes include angina, myocardial infarction, stroke or cardiovascular disease, hypertension, and death, with corresponding time-to-event or censoring variables. In our benchmark, we used death as the survival endpoint, with `DEATH` as the event indicator and `TIMEDTH` as the

observed follow-up time. Death was observed in 2,899 observations (24.9%), and 8,728 observations (75.1%) were right-censored.

FLCHAIN comes from a study of serum free light chains and mortality among residents aged 50 years or older in Olmsted County, Minnesota [12]. The final sample contains 7,874 participants selected by stratified random sampling over age and sex. We used six covariates: age, creatinine, kappa and lambda free light chain levels, MGUS status, and sex. There were 2,167 deaths (27.5%) and 5,707 censored observations (72.5%). Missing continuous values were imputed with the mean.

SUPPORT2 comes from the SUPPORT Phase II study, a multicenter study of seriously ill hospitalized adults conducted between 1992 and 1994 [6]. We used 9,105 observations with 47 covariates, including demographics, disease severity, physiologic measurements, and clinical status variables. Death was observed in 6,201 observations (68.1%), and 2,904 observations (31.9%) were right-censored. Observations with follow-up of three days or less were excluded so that baseline covariates preceded the event time.

COVID-19-NY consists of patients who tested positive for COVID-19 at Stony Brook University [41]. We used the linked clinical file and AP chest radiographs from the TCIA release. For each patient, the radiograph study closest to the hospital visit date was selected among portable/AP chest views, and the first available DICOM image from that study was converted to a grayscale PNG. The endpoint was in-hospital death, coded from discharge status, and observed time was measured in days from the selected imaging study to discharge or death. Images were padded to a square field of view and resized to 224×224 before model input.

C4KC-KiTS is a TCIA renal cancer imaging dataset with arterial-phase abdominal CT and kidney/tumor segmentations [22]. Original imaging consisted of CT DICOM series and corresponding segmentation objects. For the 2D slice selection, CT volumes were resampled to 1 mm isotropic spacing, clipped to an abdominal soft-tissue window ($[-160, 240]$ HU), and reduced to the axial slice with the largest kidney tumor cross-section according to the segmentation mask. The selected slice was resized to 224×224 and used as a single-channel grayscale image. The endpoint was overall survival after surgery: observed time was `vital_days_after_surgery`, and death was coded from `vital_status`; alive or censored patients were treated as right-censored. Only malignant cases with valid follow-up were included.

BraTS uses the BraTS 2020 glioma MRI training data [34, 2, 3]. Original imaging consisted of multi-parametric brain MRI volumes with segmentation masks; in this benchmark we used the FLAIR sequence. For the 2D slice selection, we extracted the axial FLAIR slice with the largest tumor cross-section using the segmentation mask, falling back to the center slice only if the mask was unavailable or empty. Intensities were clipped at the within-slice 99th percentile, normalized, and saved as 224×224 grayscale images. The endpoint was overall survival time in days from the BraTS survival file. We constructed the event indicator from the available vital-status/alive annotation in the processed survival file; subjects annotated as alive with follow-up time were treated as right-censored, while subjects without an alive annotation were treated as observed events.

F.2 Evaluation Horizons

Metrics were reported at three horizons: the full supported time horizon and two shorter horizons. The full horizon was chosen from the training split by estimating the censoring survival function and taking the largest time point with adequate censoring support, operationalized in the code as $\hat{G}(\tau) \geq 0.001$. The IPCW weights themselves were then capped during metric computation, as described below. The Q1 and Q2 horizons were defined from the observed time distribution of the held-out test split. These horizons were used only for reporting, not for model fitting or hyperparameter selection. In datasets with administrative censoring pile-ups, some upper quantile horizons can coincide with the full horizon; this is reported as part of the dataset behavior rather than manually adjusted.

F.3 Evaluation Metrics

Time-dependent C-index. We used an IPCW version of the time-dependent C-index to evaluate discrimination under right censoring [46]. Let $\hat{S}_i(t) = \hat{S}_\theta(t|x_i)$ be the predicted survival probability for subject i, and let $\hat{G}(t) = P(C > t)$ be the censoring survival function estimated by Kaplan-Meier on the training split. Comparable pairs are formed from subjects with observed events before the

evaluation horizon. A pair is concordant when the subject who fails earlier is assigned a lower predicted survival probability at that event time.

$$C_\tau^{td} = \frac{\sum_{i,\delta_i=1,o_i<\tau} \sum_{j:o_i<o_j} \mathbb{I}\{\hat{S}_i(o_i) < \hat{S}_j(o_i)\} w_i}{\sum_{i,\delta_i=1,o_i<\tau} \sum_{j:o_i<o_j} w_i}, \qquad w_i = \min\{1/\hat{G}^2(o_i), 10\}.$$

The cap at 10 was used for all IPCW-based metrics. It is a numerical stabilization step, not a subject-level exclusion rule. Subjects were not removed simply because they occurred late in follow-up.

Integrated Brier Score. The Brier score measures the squared error between predicted survival and the observed survival status at a fixed time. We used the IPCW form proposed for right-censored survival data [18]. At time t,

$$\mathrm{BS}(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\big(\hat{S}_i(t)\big)^2 \mathbb{I}(o_i < t, \delta_i = 1)}{\hat{G}(o_i)} + \frac{\big(1 - \hat{S}_i(t)\big)^2 \mathbb{I}(o_i > t)}{\hat{G}(t)} \right],$$

with the same IPCW cap applied to the inverse censoring weights. The integrated Brier score (IBS) was computed by numerical integration over the evaluation interval. Lower IBS indicates better calibrated survival prediction.

Integrated binomial log-likelihood. We also report an IPCW binomial log-likelihood, which is sensitive to both calibration and discrimination. At time t,

$$\mathrm{BLL}(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\log\big(1 - \hat{S}_i(t)\big) \mathbb{I}(o_i < t, \delta_i = 1)}{\hat{G}(o_i)} + \frac{\log\big(\hat{S}_i(t)\big) \mathbb{I}(o_i > t)}{\hat{G}(t)} \right].$$

The integrated version, denoted IBLL, was computed over the same time grid as IBS. Because this is a log-likelihood rather than a loss, higher values are better.

D-calibration. We also report distributional calibration (D-calibration), following [19]. D-calibration assesses whether the model's predicted individual survival distributions are calibrated over the full follow-up range, rather than at a single fixed time point. Under a correctly specified continuous survival model, the probability integral transform implies that $F(T|X)$ is uniformly distributed on $[0, 1]$; equivalently, $S(T|X)$ is also uniformly distributed on $[0, 1]$. Thus, for uncensored subjects, we evaluate the predicted survival probability at the observed event time, $\hat{S}_i(o_i)$.

Right-censoring requires a partial contribution because the true event time is only known to exceed o_i. On the survival probability scale, the unobserved value $\hat{S}_i(T_i)$ must lie in the interval $[0, \hat{S}_i(o_i)]$. Following the standard D-calibration construction, each censored subject's contribution is therefore distributed uniformly over this compatible interval. We divide $[0, 1]$ into 10 equal-width bins and compare the resulting bin counts with the uniform distribution using a chi-square goodness-of-fit statistic with 9 degrees of freedom. We report the corresponding p-value, where larger values indicate less evidence against distributional calibration. Because this is a hypothesis-test diagnostic rather than an effect-size metric, we use D-calibration as a secondary calibration assessment rather than as the primary model-ranking criterion.

F.4 Average Ranks and Paired Tests

Average ranks in Table 2 are ranks of the mean performance within each dataset and horizon, averaged over datasets. All methods share the same data splits, so we compare QSurv with each baseline by a two-sided paired t-test over the matched splits. Table 8 reports the mean paired difference at the full horizon, with the half-width of its 95% confidence interval.

F.5 Implementation Details

Tabular Data

For tabular datasets, all models used a multilayer perceptron (MLP) backbone to extract fixed-dimensional representations from input covariates. The MLP consists of multiple hidden layers with

Table 8: Mean paired difference between QSurv and each baseline at the full horizon ± the half-width of its 95% t-interval over the paired splits. Δ is QSurv − baseline for C_{td} and IBLL and baseline − QSurv for IBS, so $\Delta > 0$ favors QSurv for every metric. A bold difference has an interval that excludes zero.

Dataset	Baseline	C_{td}	IBS	IBLL
COVID-19-NY	CoxCC	**+0.1763** ± 0.0399	**+0.1169** ± 0.1051	**+0.2893** ± 0.2633
	CoxTime	**+0.1869** ± 0.1243	+0.1363 ± 0.1537	+0.3879 ± 0.4666
	DeepHit	**+0.0780** ± 0.0349	-0.0008 ± 0.0340	+0.0108 ± 0.0950
	NnetSurv	+0.0796 ± 0.1265	+0.0081 ± 0.0312	+0.0311 ± 0.0864
	MDN	+0.0568 ± 0.1404	**+0.1176** ± 0.0999	**+0.3308** ± 0.3093
	DeSurv	+0.0036 ± 0.0435	-0.0100 ± 0.0294	-0.0337 ± 0.0861
	SODEN	+0.0229 ± 0.0771	+0.0019 ± 0.0227	+0.0568 ± 0.1192
C4KC-KiTS	CoxCC	**+0.2265** ± 0.0867	-0.0161 ± 0.0548	-0.0358 ± 0.1140
	CoxTime	+0.0573 ± 0.2412	-0.0174 ± 0.0570	-0.0431 ± 0.1240
	DeepHit	+0.1292 ± 0.2293	+0.0919 ± 0.1250	+0.3629 ± 0.4578
	NnetSurv	+0.0959 ± 0.2078	+0.0002 ± 0.0546	+0.1228 ± 0.3180
	MDN	+0.1151 ± 0.2532	-0.0154 ± 0.0538	+0.0740 ± 0.2676
	DeSurv	+0.0641 ± 0.1626	-0.0104 ± 0.0583	+0.2106 ± 0.4964
	SODEN	+0.0808 ± 0.3545	+0.0092 ± 0.0718	+0.9280 ± 1.2682
BraTS	CoxCC	**+0.0886** ± 0.0666	+0.0983 ± 0.1539	+0.2895 ± 0.4530
	CoxTime	**+0.1270** ± 0.0410	**+0.0133** ± 0.0128	**+0.0438** ± 0.0411
	DeepHit	+0.0615 ± 0.0891	+0.0126 ± 0.0221	+0.0439 ± 0.0542
	NnetSurv	+0.0455 ± 0.0833	+0.0205 ± 0.0427	+0.1759 ± 0.4237
	MDN	**+0.0477** ± 0.0369	**+0.0235** ± 0.0097	**+0.0804** ± 0.0298
	DeSurv	+0.0352 ± 0.0554	-0.0000 ± 0.0066	+0.0078 ± 0.0436
	SODEN	+0.0305 ± 0.0794	+0.0016 ± 0.0084	+0.0182 ± 0.0593
FLCHAIN	CoxCC	-0.0002 ± 0.0009	+0.0000 ± 0.0003	+0.0000 ± 0.0011
	CoxTime	+0.0007 ± 0.0013	+0.0005 ± 0.0004	+0.0017 ± 0.0024
	DeepHit	**+0.0083** ± 0.0018	**+0.0121** ± 0.0011	**+0.0297** ± 0.0029
	NnetSurv	**+0.0078** ± 0.0020	-0.0000 ± 0.0005	+0.0000 ± 0.0018
	MDN	+0.0011 ± 0.0016	**+0.0011** ± 0.0005	**+0.0040** ± 0.0018
	DeSurv	-0.0002 ± 0.0010	+0.0001 ± 0.0003	+0.0003 ± 0.0010
FRAMINGHAM	CoxCC	+0.0013 ± 0.0018	-0.0003 ± 0.0004	-0.0008 ± 0.0013
	CoxTime	+0.0017 ± 0.0026	-0.0001 ± 0.0005	+0.0001 ± 0.0023
	DeepHit	**+0.0159** ± 0.0032	**+0.0896** ± 0.0071	**+0.2283** ± 0.0177
	NnetSurv	**+0.0104** ± 0.0032	**+0.0017** ± 0.0009	**+0.0061** ± 0.0034
	MDN	**+0.0042** ± 0.0028	+0.0005 ± 0.0006	+0.0023 ± 0.0024
	DeSurv	+0.0007 ± 0.0020	-0.0002 ± 0.0006	-0.0005 ± 0.0018
GBSG	CoxCC	+0.0042 ± 0.0110	-0.0002 ± 0.0043	+0.0003 ± 0.0109
	CoxTime	+0.0073 ± 0.0129	**+0.0065** ± 0.0062	**+0.0174** ± 0.0166
	DeepHit	**+0.0261** ± 0.0112	**+0.0311** ± 0.0135	**+0.0757** ± 0.0344
	NnetSurv	**+0.0352** ± 0.0166	+0.0241 ± 0.0338	+0.2311 ± 0.4386
	MDN	+0.0067 ± 0.0105	**+0.0058** ± 0.0046	**+0.0188** ± 0.0132
	DeSurv	+0.0061 ± 0.0078	+0.0004 ± 0.0031	+0.0007 ± 0.0073
METABRIC	CoxCC	**+0.0205** ± 0.0088	-0.0008 ± 0.0037	+0.0014 ± 0.0118
	CoxTime	-0.0038 ± 0.0082	+0.0002 ± 0.0024	+0.0009 ± 0.0062
	DeepHit	-0.0075 ± 0.0085	**+0.0051** ± 0.0033	**+0.0179** ± 0.0091
	NnetSurv	**+0.0092** ± 0.0087	**+0.0068** ± 0.0029	**+0.0205** ± 0.0086
	MDN	**+0.0186** ± 0.0056	**+0.0150** ± 0.0034	**+0.0479** ± 0.0094
	DeSurv	-0.0031 ± 0.0073	-0.0010 ± 0.0022	-0.0031 ± 0.0060
NWTCO	CoxCC	+0.0073 ± 0.0097	+0.0007 ± 0.0013	+0.0024 ± 0.0047
	CoxTime	+0.0035 ± 0.0082	+0.0004 ± 0.0011	+0.0012 ± 0.0039
	DeepHit	**+0.0136** ± 0.0070	**+0.0028** ± 0.0010	**+0.0080** ± 0.0038
	NnetSurv	**+0.0195** ± 0.0075	**+0.0036** ± 0.0013	**+0.0106** ± 0.0048
	MDN	-0.0036 ± 0.0065	**-0.0016** ± 0.0008	**-0.0043** ± 0.0031
	DeSurv	-0.0001 ± 0.0039	-0.0004 ± 0.0006	-0.0019 ± 0.0022
SUPPORT2	CoxCC	**+0.0167** ± 0.0026	+0.0022 ± 0.0022	**+0.0139** ± 0.0096
	CoxTime	+0.0009 ± 0.0023	+0.0006 ± 0.0017	+0.0036 ± 0.0043
	DeepHit	**+0.0090** ± 0.0024	**+0.0138** ± 0.0013	**+0.0342** ± 0.0035
	NnetSurv	**+0.0215** ± 0.0021	-0.0003 ± 0.0013	-0.0017 ± 0.0039
	MDN	**+0.0191** ± 0.0027	**+0.0109** ± 0.0019	**+0.0254** ± 0.0043
	DeSurv	-0.0021 ± 0.0027	+0.0009 ± 0.0019	+0.0033 ± 0.0051

nonlinear activations, dropout, and optional batch normalization. Within each experiment, all models used the same backbone search space, followed by model-specific output heads determined by each survival modeling formulation.

We performed hyperparameter optimization using random search. Each candidate configuration was trained for up to 200 epochs, and the final configuration was selected based on the best validation C_{td}, with negative validation IBS used as a tie-breaker. We used AdamW as the optimizer with a cosine learning-rate schedule, setting $T_{\max}$ to the maximum number of epochs and $\eta_{\min} = 0$. For each tabular dataset and model, we evaluated 30 random hyperparameter configurations. The search space is summarized in Table 9. For NnetSurv and DeepHit, the number of discrete time bins was

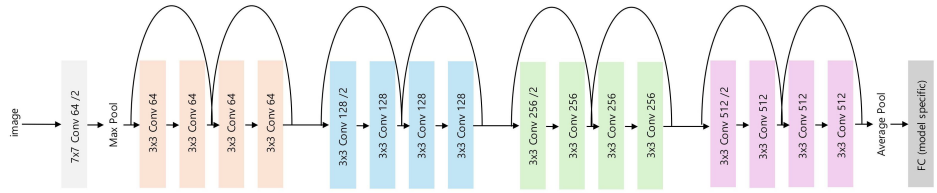

Figure 5: ResNet-18 backbone architecture used for medical imaging survival modeling. The original fully connected layer is removed, and model-specific heads are applied to the extracted feature vectors to produce survival predictions.

fixed at 50. For MDN, the number of mixture components was fixed at 5. QSurv used $K = 15$ Gauss-Legendre nodes in all real-data experiments.

Table 9: Hyperparameter search space for tabular datasets.

Hyperparameter	Search space
Number of dense hidden layers	$\{2, 3, 4\}$
Hidden size	$\{32, 64, 128, 256\}$
Learning rate	LogUniform$(10^{-4}, 10^{-2})$
Weight decay	LogUniform$(10^{-8}, 10^{-3})$
Dropout	$\{0.0, 0.1, 0.3, 0.5\}$
Batch size	$\{64, 128, 256\}$
Batch normalization	$\{\texttt{True}, \texttt{False}\}$

Medical Imaging Data

For medical imaging datasets, all models used ResNet-18 as the convolutional backbone. The original fully connected layer was removed, and the extracted feature vector was passed to a model-specific survival head. The first convolutional layer was modified when the input channel dimension differed from the standard three-channel RGB setting. For single-channel inputs with pretrained weights, the first-layer filters were initialized by averaging the pretrained RGB filters across channels. As shown in Figure 5, all image-based models shared the same ResNet-18 feature extractor, while the final prediction heads differed according to each model's output parameterization.

We performed random-search hyperparameter optimization for medical imaging models using the search space in Table 10. Each candidate configuration was trained for up to 200 epochs, and the final configuration was selected using validation C_{td}, with negative validation IBS used as a tie-breaker. We used AdamW with a cosine learning-rate schedule, again setting $T_{\max}$ to the maximum number of epochs and $\eta_{\min} = 0$. For each imaging dataset and model, we evaluated 20 random hyperparameter configurations. Hidden-layer dimensions were not searched because the feature dimension was fixed by the ResNet-18 backbone. As in the tabular experiments, the number of time bins was fixed at 50 for NnetSurv and DeepHit, and the number of mixture components was fixed at 5 for MDN.

Table 10: Hyperparameter search space for medical imaging datasets. Hidden-layer dimensions are not searched because the feature dimension is fixed by the convolutional backbone.

Hyperparameter	Search space
Learning rate	LogUniform$(10^{-5}, 10^{-2})$
Weight decay	LogUniform$(10^{-8}, 10^{-3})$
Dropout	$\{0.0, 0.1, 0.3, 0.5\}$
Batch size	$\{16, 32, 64\}$

Table 11: Training throughput, peak GPU memory, and backbone and hazard evaluations per epoch on a single NVIDIA A100 80GB GPU, with the same backbone, Time-LoRA head, batch size and number of epochs for all three models. QSurv and DeSurv use $K = 15$ Gauss-Legendre nodes and SODEN uses an adaptive dopri5 solver with adjoint backpropagation.

Dataset	Model	Throughput (samples/s)	Peak memory (MiB)	Backbone evaluations	Hazard evaluations
COVID-19-NY	DeSurv	393 ± 40	1,650.3 ± 0.0	70 ± 0	17,248 ± 0
	SODEN	14 ± 0	2,297.5 ± 0.0	2,690 ± 25	83,199 ± 809
	QSurv	397 ± 27	1,650.0 ± 0.0	70 ± 0	17,248 ± 0
C4KC-KiTS	DeSurv	299 ± 143	1,650.2 ± 0.0	12 ± 0	2,336 ± 0
	SODEN	13 ± 1	2,295.6 ± 0.0	456 ± 17	11,686 ± 455
	QSurv	389 ± 3	1,649.9 ± 0.0	12 ± 0	2,336 ± 0
BraTS	DeSurv	306 ± 142	1,649.6 ± 0.0	14 ± 0	3,008 ± 0
	SODEN	13 ± 1	2,296.7 ± 0.0	534 ± 8	14,669 ± 227
	QSurv	325 ± 134	1,649.3 ± 0.0	14 ± 0	3,008 ± 0
FLCHAIN	DeSurv	14,572 ± 336	67.5 ± 0.0	100 ± 0	100,736 ± 0
	SODEN	614 ± 12	71.1 ± 0.0	3,715 ± 37	470,327 ± 4,766
	QSurv	15,717 ± 259	67.3 ± 0.0	100 ± 0	100,736 ± 0
FRAMINGHAM	DeSurv	14,054 ± 32	67.5 ± 0.0	56 ± 0	54,688 ± 0
	SODEN	558 ± 8	71.5 ± 0.0	2,161 ± 21	263,663 ± 2,390
	QSurv	15,239 ± 38	67.4 ± 0.0	56 ± 0	54,688 ± 0
GBSG	DeSurv	11,400 ± 59	67.5 ± 0.0	12 ± 0	8,768 ± 0
	SODEN	498 ± 10	71.2 ± 0.0	411 ± 7	39,276 ± 715
	QSurv	12,311 ± 91	67.3 ± 0.0	12 ± 0	8,768 ± 0
METABRIC	DeSurv	14,176 ± 375	67.5 ± 0.0	24 ± 0	24,352 ± 0
	SODEN	632 ± 15	71.2 ± 0.0	823 ± 10	104,420 ± 1,316
	QSurv	15,446 ± 254	67.3 ± 0.0	24 ± 0	24,352 ± 0
NWTCO	DeSurv	14,121 ± 96	67.5 ± 0.0	52 ± 0	51,552 ± 0
	SODEN	472 ± 11	71.1 ± 0.0	2,318 ± 42	290,558 ± 5,297
	QSurv	15,123 ± 328	67.3 ± 0.0	52 ± 0	51,552 ± 0
SUPPORT2	DeSurv	14,497 ± 28	67.6 ± 0.0	116 ± 0	114,768 ± 0
	SODEN	401 ± 4	71.8 ± 0.1	5,958 ± 56	745,268 ± 7,318
	QSurv	15,684 ± 180	67.4 ± 0.0	116 ± 0	114,768 ± 0

F.6 Computational Resources

All experiments are reproducible on standard CPU or GPU hardware. Tabular experiments were run on CPU only and require a single node with at least 24 GB of system memory. Each hyperparameter optimization cell, corresponding to one method, one dataset, and one seed with 30 trials and up to 200 training epochs, takes approximately 5–30 minutes of wall-clock time depending on dataset size. The full tabular benchmark, consisting of 6 datasets, 7 methods, and 20 seeds, requires approximately 270 CPU-hours.

Image experiments require a single GPU with at least 20 GB of memory. We used NVIDIA L40S 44 GB and A100 80 GB nodes interchangeably. Each hyperparameter optimization cell, consisting of 20 trials and up to 200 training epochs, takes approximately 15-90 minutes depending on dataset size. The full image benchmark, consisting of 3 datasets, 8 methods, and 5 seeds, requires approximately 90 GPU-hours.

The synthetic simulation study can be run on a laptop-class CPU. In our experiments, the full simulation sweep was run on an Apple M4 machine with 10 CPU cores and 32 GB RAM and completed in approximately 6-8 hours. SODEN with adjoint backpropagation dominated the simulation wall-clock time, requiring approximately 130 seconds per fit, whereas all other methods completed in approximately 1-7 seconds per fit.

Including preliminary experiments, repeated hyperparameter-configuration runs, and debugging runs that were not included in the final reported results, the total compute used for the project is estimated to be approximately 3–5 times the headline compute reported above.

F.7 Deep Survival Analysis Models

CoxCC and CoxTime CoxCC and CoxTime extend the Cox proportional hazards framework by replacing the linear predictor with a neural network [32]. CoxCC retains the proportional hazards

assumption,

$$\lambda(t|x) = \lambda_0(t) \exp\{g(x)\}, \tag{19}$$

where $\lambda_0(t)$ is the baseline hazard and $g(x)$ is a time-independent neural risk score. CoxTime relaxes this assumption by allowing the risk score to depend on time,

$$\lambda(t|x) = \lambda_0(t) \exp\{g(x,t)\}, \tag{20}$$

thereby permitting time-varying covariate effects and crossing survival curves. We used the PyCox implementation: `https://github.com/havakv/pycox`.

To reduce the computational cost of evaluating full risk sets, these implementations use a case-control approximation to the Cox partial likelihood. For a subject i who experiences an event at time T_i, the model compares the event subject against a sampled control set J_i of individuals who remain at risk at T_i. The resulting loss is

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|J_i|} \sum_{j \in J_i} \log\left(1 + \exp\{g(x_j) - g(x_i)\}\right). \tag{21}$$

This softplus form penalizes the model when a control subject j is assigned a higher risk score than the event subject i.

After training, a baseline hazard estimate is required to obtain absolute survival probabilities. Since the neural network predicts relative risk, the baseline hazard is recovered nonparametrically using the Breslow estimator. For a unique event time t_m with d_m events and risk set $\mathcal{R}(t_m)$, the CoxCC baseline hazard estimate is

$$\hat{\lambda}_0(t_m) = \frac{d_m}{\sum_{j \in \mathcal{R}(t_m)} \exp\{g(x_j)\}}. \tag{22}$$

For CoxTime, the risk score depends on the event time, so the network must be evaluated at t_m for each subject in the risk set:

$$\hat{\lambda}_0(t_m) = \frac{d_m}{\sum_{j \in \mathcal{R}(t_m)} \exp\{g(x_j, t_m)\}}. \tag{23}$$

The cumulative baseline hazard $\hat{\Lambda}_0(t)$ is then obtained by summing these estimates over event times $t_m \leq t$.

DeepHit DeepHit formulates survival analysis as a discrete-time prediction problem [33]. The time axis is divided into intervals, and the network outputs a probability mass function $\mathbf{p} = (p_1, \ldots, p_K)$, where p_k is the predicted probability that the event occurs in interval k. The cumulative incidence function is

$$F(t_k|x) = \sum_{j=1}^{k} p_j. \tag{24}$$

We implemented DeepHit based on the PyCox implementation: `https://github.com/havakv/pycox`.

The training objective combines a likelihood term and a ranking loss:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{NLL}} + \alpha \mathcal{L}_{\text{Rank}}, \tag{25}$$

where α controls the relative weight of the two components. For an uncensored subject whose event occurs in interval k^*, the likelihood contribution is $-\log p_{k^*}$. For a subject censored in interval k^*, the likelihood contribution is $-\log\{1 - F(t_{k^*}|x)\}$. The ranking loss encourages subjects who experience earlier events to have higher predicted cumulative incidence at the corresponding event time:

$$\mathcal{L}_{\text{Rank}} = \sum_{A,B} \mathbb{I}(T_A < T_B) \exp\left(\frac{F(T_A|x_B) - F(T_A|x_A)}{\sigma}\right), \tag{26}$$

where σ is a smoothing parameter.

NnetSurv NnetSurv models survival in discrete time through interval-specific conditional event probabilities. Let $\lambda_j(x)$ denote the probability of an event in interval $[t_{j-1}, t_j)$ conditional on survival up to t_{j-1}:

$$\lambda_j(x) = P(T \in [t_{j-1}, t_j) | T \geq t_{j-1}, x) = \sigma(f_j(x)). \tag{27}$$

The corresponding survival probability at t_k is

$$S(t_k|x) = \prod_{j=1}^{k} \{1 - \lambda_j(x)\}. \tag{28}$$

The model is trained with a binary cross-entropy loss over all intervals in which each subject is at risk. For subject i with observed interval index k_i and event indicator δ_i, the target is $y_{ij} = 0$ for $j < k_i$ and $y_{ik_i} = \delta_i$. The loss is

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{j=1}^{k_i} \left[y_{ij} \log \lambda_j(x_i) + (1 - y_{ij}) \log\{1 - \lambda_j(x_i)\} \right]. \tag{29}$$

Survival Mixture Density Network The Survival Mixture Density Network (MDN) models the time-to-event distribution using a Gaussian mixture distribution on a latent time scale [20]. We implemented MDN based on the authors' official codebase: `https://github.com/XintianHan/Survival-MDN`. To ensure positive event times, the observed time is represented as $T = \text{softplus}(T^*) = \log(1 + \exp\{T^*\})$, where the latent variable T^* follows a conditional Gaussian mixture:

$$f_{T^*}(u|x) = \sum_{k=1}^{K} \pi_k(x) \mathcal{N}\{u; \mu_k(x), \sigma_k(x)\}. \tag{30}$$

The density of the observed event time follows from the change-of-variables formula:

$$f_T(t|x) = f_{T^*}\{\text{softplus}^{-1}(t)|x\} \left| \frac{d}{dt} \text{softplus}^{-1}(t) \right| = f_{T^*}\{\text{softplus}^{-1}(t)|x\} \frac{\exp(t)}{\exp(t) - 1}. \tag{31}$$

The survival function is

$$S_T(t|x) = 1 - F_{T^*}\{\text{softplus}^{-1}(t)|x\}, \tag{32}$$

where F_{T^*} is the CDF of the latent Gaussian mixture. The model is trained by minimizing the censored negative log-likelihood,

$$\mathcal{L} = -\sum_{i=1}^{n} \left[\delta_i \log f_T(t_i|x_i) + (1 - \delta_i) \log S_T(t_i|x_i) \right]. \tag{33}$$

DeSurv DeSurv parameterizes the conditional distribution function through a monotone neural construction [10]. Rather than modeling the hazard directly, DeSurv defines a non-negative function

$$g(t, x) = \frac{\partial u(t, x)}{\partial t} \tag{34}$$

using a softplus output, and constructs

$$u(t, x) = \int_0^t g(s, x) ds. \tag{35}$$

In our implementation, this integral is evaluated using fixed Gauss–Legendre quadrature with 15 nodes. The conditional CDF is then obtained by mapping $u(t, x)$ to $[0, 1]$:

$$F(t|x) = \tanh\{u(t, x)\}. \tag{36}$$

The event density follows by differentiation:

$$f(t|x) = \frac{\partial F(t|x)}{\partial t} = \left[1 - F(t|x)^2\right] g(t, x). \tag{37}$$

Thus,

$$S(t|x) = 1 - F(t|x), \qquad \lambda(t|x) = \frac{f(t|x)}{S(t|x)} = \{1 + F(t|x)\} g(t, x). \tag{38}$$

The hazard is therefore a derived quantity rather than the primitive parameterized object. DeSurv is trained by minimizing the standard censored negative log-likelihood,

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \left[\delta_i \log f(t_i|x_i) + (1 - \delta_i) \log S(t_i|x_i) \right]. \tag{39}$$

SODEN SODEN is a continuous-time survival model based on Neural Ordinary Differential Equations [44]. We implemented SODEN using the authors' codebase: `https://github.com/jiaqima/SODEN`. The model represents the instantaneous hazard $\lambda(t|x)$ with a neural network and computes the cumulative hazard through the ODE integral

$$\Lambda(t|x) = \int_0^t \lambda(u|x)du. \tag{40}$$

This integral is evaluated during the forward pass using a differentiable adaptive ODE solver. The model is trained by minimizing the censored negative log-likelihood,

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \left[\delta_i \log \lambda(t_i|x_i) - \Lambda(t_i|x_i)\right]. \tag{41}$$

If the input time scale is transformed during preprocessing, the corresponding Jacobian adjustment is included so that the likelihood remains valid on the original time scale.

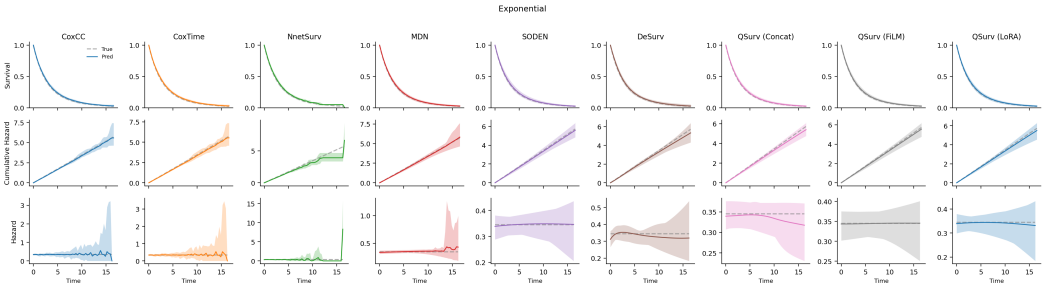

Figure 6: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for exponential distribution scenario. Shades are 95% confidence interval.

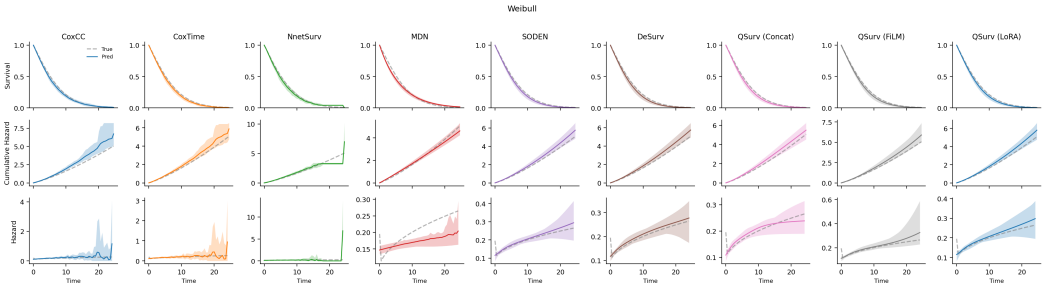

Figure 7: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for Weibull distribution scenario. Shades are 95% confidence interval.

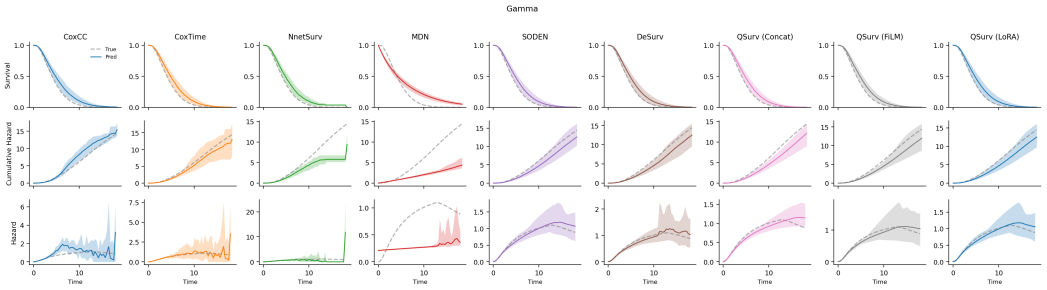

Figure 8: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for gamma distribution scenario. Shades are 95% confidence interval.

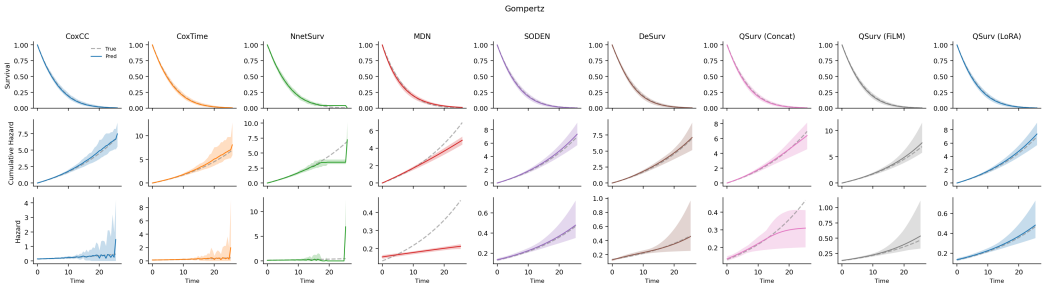

Figure 9: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for Gompertz distribution scenario. Shades are 95% confidence interval.

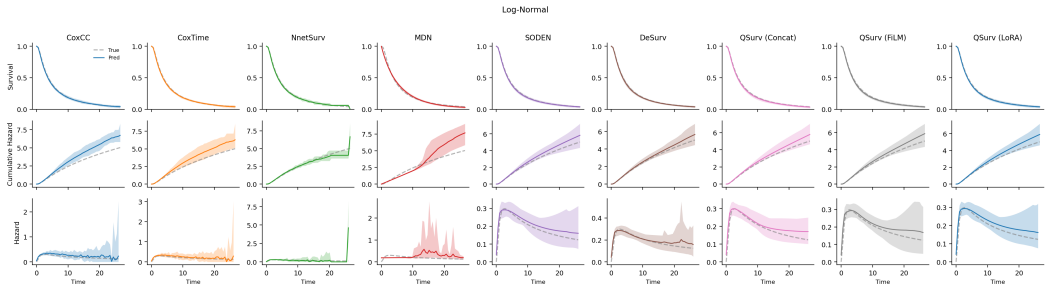

Figure 10: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for log-normal distribution scenario. Shades are 95% confidence interval.

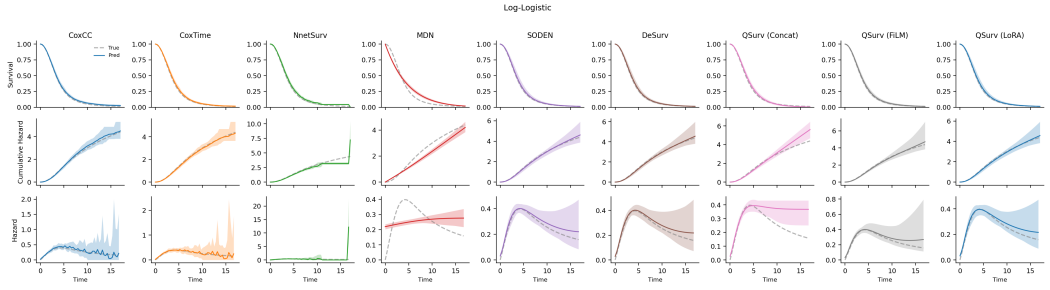

Figure 11: True vs. predicted survival, cumulative hazard, and instantaneous hazard functions for log-logistic distribution scenario. Shades are 95% confidence interval.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the main contributions and scope of the paper. The claims are consistent with the proposed method, theoretical motivation, and experimental evaluation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the scope of the proposed approach in Section 4.2 and its limitations in Section 5. Computational cost is reported in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical results are stated with their relevant assumptions, and complete proofs are provided in Appendix B. The results are numbered and cross-referenced in the main text.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the proposed model, training procedure, datasets, baselines, evaluation metrics, and hyperparameter settings in sufficient detail to reproduce the main experimental results. Additional implementation and experimental details are provided in Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code for QSurv, the baselines and the evaluation is available at `https://github.com/hyungrok-do/QSurv`, with instructions for data access and preprocessing, model training, baseline comparison, and evaluation. All datasets are publicly available from the sources cited in Appendix F.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental settings needed to understand the results, including training procedures, optimizer choices, hyperparameters, baseline implementations, and evaluation metrics. Additional details are provided in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports the mean and standard deviation over 20 repetitions for the tabular datasets and 5 repetitions for the imaging datasets, each with a new random data split. Paired t-tests and 95% confidence intervals over the matched splits are reported in Appendix F.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F.6 reports the compute resources required for each experiment, including CPU or GPU requirements, memory requirements, approximate runtime per experiment, and total estimated compute for the tabular, image, and simulation benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The experiments use appropriate benchmark data and evaluation procedures, and the paper does not involve deceptive practices, harmful deployment, or inappropriate use of sensitive information.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper proposes a general methodological framework for survival modeling and does not directly deploy a system or make decisions about individuals. Therefore, the work has no direct societal impact by itself; potential impacts would depend on downstream applications and would require separate validation, fairness analysis, and appropriate domain-specific safeguards.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases no data or models with a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper credits the original sources of the datasets, code packages, and baseline methods used in the experiments. Dataset access conditions, licenses, and terms of use are described where applicable, and all assets are used in accordance with their intended research use settings.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper introduces no new datasets or models as assets. The released code is documented in its repository.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper involves no crowdsourcing or research with human subjects. It uses only existing public benchmark datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no new study participants. All analyses use existing de-identified public datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods do not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.