DeepHazard: A Tensorflow/Keras and PyTorch-Compatible Deep Learning Package for Survival Analysis

Gianmarco Sabbatini* and Lorenzo Manganaro

aizoOn Technology & Consulting s.r.l. Strada del Lionetto, 6, 10146, Torino (TO), Italy

Abstract

Survival analysis is a statistical approach that studies time-to-event data, with applications in many fields ranging from medicine to social sciences. While deep learning has emerged as a powerful tool for survival analysis, existing software solutions often lack flexibility, rely on unmaintained frameworks, or do not support key functionalities such as transfer learning. To address these limitations, we present DeepHazard, a novel Python package designed to facilitate the implementation of deep neural networks for survival analysis. Built with compatibility for both TensorFlow/Keras and PyTorch, DeepHazard provides a customizable architecture, efficient handling of censored data, pre-defined loss function derived from the Cox model and including both L1/L2 regularization, and built-in support for transfer learning. Representing a versatile and user-friendly framework, DeepHazard enables researchers to easily develop, customize, and deploy deep learning models for survival analysis and risk stratification with applications in biomedical sciences and beyond. The package is designed for ease of use, reproducibility, and integration into modern machine learning workflows.

Keywords

Survival analysis, Deep learning, Time-to-event prediction, Tensorflow, Keras, PyTorch.

1. Introduction

Survival analysis, also referred to as time-to-event analysis, is a branch of statistics that studies the amount of time it takes before a particular event of interest occur [1]. Depending on the type of event considered, it has a wide range of applications in many different fields, such as actuarial sciences, that rely on survival models to predict life expectancy and design insurance policies [2], social sciences, in which it is used for example to study career progression and job retention [3], or industrial application, in which this kind of analysis is widely used for predictive maintenance [4], that is the attempt of anticipating equipment failure and strategically planning maintenance or replacements of key components of industrial plants.

A unique feature of this kind of analysis with respect to conventional regression, is the need of handling censored data, that primarily arise because of incomplete follow-up of the samples. In particular, censoring occurs when the event of interest is not yet happened at the time of analysis [1]. This makes survival analysis particularly well-suited for biomedical applications, where the event of interest can be death, disease recurrence, or any other clinically relevant outcome, and in which patients may have not yet experienced such event during the course of a clinical trial, or the outcomes may be unknown at the time of analysis, for example because the patient withdrew from the study. In fact, in this field, survival analysis has been widely used to study patient prognosis, evaluate treatment effectiveness, and develop risk prediction models [5].

In the last decades, with the advent of precision and personalized medicine, the need for robust survival modelling techniques became increasingly evident. In fact, precision medicine extensively exploit predictive algorithms to personalize the treatment strategies based on individual genetic

2nd Workshop "New frontiers in Big Data and Artificial Intelligence" (BDAI 2025), May 29-30, 2025, Aosta, Italy
*Corresponding author.
gianmarco.sabbatini@aizoongroup.com (G. Sabbatini); lorenzo.manganaro@aizoongroup.com (L. Manganaro)
0000-0003-4755-1638 (L. Manganaro)

profiles [6], systematically acquired thanks to high-throughput and cost-effective technologies such as next generation sequencing (NGS) and its evolutions [7].

However, leveraging these vast, heterogeneous datasets to get reliable survival predictions poses a significant challenge for conventional statistical models, such as the Cox proportional hazards model [8], that often rely on strong assumptions about the relationships between covariates and survival, that may not hold in complex biological systems, and struggle with the high-dimensional and strongly non-linear nature of genomic and multi-omic data.

Instead, this is where deep learning is rapidly emerging as a powerful and robust alternative [9]. In fact, first of all it is intrinsically designed to catch non-linear patterns and correlations within the data. Secondly, by exploiting many flexible and diverse architectures, ranging from deep artificial neural networks [10], to convolutional [11] or recurrent [12] networks, it can be easily adapted to handle and integrate different data sources, from clinical data, to medical imaging or molecular data [13]. Moreover, deep learning models can be optimized for transfer learning [14], in which knowledge gained from larger datasets can be transferred and applied to smaller or domain-specific cohorts, an essential feature given the limited number of patients or samples characterizing many clinical datasets.

In this context, there is an urgent need of frameworks designed to facilitate the development of deep learning models for survival analysis applications.

1.1. Related works

In the last few years, several deep learning-based libraries have been developed to address this need. However, all the existing tools present limitations that we aim to overcome with the present work. A detailed review of existing deep learning methods for survival analysis is beyond the scope of this paper and can be found elsewhere [15]; here, we limit to provide a selected overview of the most relevant existing packages:

1. DeepSurv [16]: provides a deep learning extension of the Cox proportional hazard model, implemented using Theano [17] and Lasagne [18]. DeepSurv has demonstrated improved predictive performance over traditional Cox models by capturing complex, non-linear relationships within the data [16]. However, it lacks a built-in support for transfer learning applications. Moreover, and most importantly, it is entirely based on Theano, which (beside being considered less user-friendly, due to its lower-level nature) is no longer maintained, representing a huge limitation with respect to modern deep learning frameworks such as Tensorflow/Keras and PyTorch.
2. Cox-nnet [19] / Cox-nnet v2.0 [20]: Similar to DeepSurv, Cox-nnet provides a framework for the implementation of a deep neural network for survival analysis that extends the Cox proportional hazard model. Cox-nnet was developed with a specific focus on gene expression datasets, despite its implementation has no differences with respect to a general purpose network. With respect to DeepSurv, Cox-nnet provides less flexibility in terms of regularization, since it does not provide the option of including dropout layers and it only implements the Ridge penalization method that may be suboptimal for many datasets. Moreover, Cox-nnet is also built on Theano, leading to similar concerns regarding maintainability and compatibility with current deep learning ecosystems.
3. DeepHit [21]: This method addresses survival analysis within the context of competing risks, that is only appropriate when there is the need to account for multiple mutually exclusive event types, so that the goal is to directly estimate the probability of different events over time. While DeepHit offers a flexible approach to modeling multiple event types, its focus diverges from traditional survival analysis tasks that instead involve single-event outcomes. While DeepHit is built based on Tensorflow/Keras that represents a gold standard for deep learning applications, it does not integrate transfer learning functionalities, limiting its applicability in the case of small datasets.

4. PyCox [22]: is a Python package devoted to survival analysis: It is built on PyTorch, thus benefiting of its flexibility and active development. PyCox is primarily a collection of existing models, including DeepSurv and DeepHit, but it does not introduce novel functionalities such as customized cost functions or native support for transfer learning applications.
5. TorchSurv [23]: is a PyTorch-based package for survival analysis developed by Novartis in collaboration with the US Food and Drug Administration. The goal of the package is to provide an intuitive tool for implementing deep neural networks for survival analysis. It implements loss functions derived from both parametric (Weibull accelerated failure time) and semi-parametric (Cox proportional hazard) models, without the inclusion of any additional regularization terms. While it a re-implements several well-established evaluation metrics (typically provided by independent packages, such as scikit-survival), but it does not provide built-in functionality for transfer learning applications.
6. TorchLife (https://github.com/sachinruk/torchlife/): Similar to TorchSurv, but with restricted functionalities, TorchLife is a PyTorch-based package with a minimal implementation of deep learning based models for survival analysis. It shares the same limitations already identified in the case of TorchSurv.
7. Auton-survival [24]: is a PyTorch-based package extending both proportional and time-dependent hazard models to deep learning architectures. Besides standard survival analysis approaches, and unlike the packages, it also provides useful functionality for counterfactual estimation and supervised or unsupervised risk stratification. However, similar to the other packages, it lacks support for transfer learning applications.

1.2. Objective and structure of the paper

The objective of this paper is to present DeepHazard, a novel python package for the implementation of deep learning models devoted to survival analysis that extends the regularized Cox proportional hazard model. The package aims to address the limitations of the existing packages, described in section 1.1, by providing a versatile and user-friendly solution both compatible with Tensorflow/Keras and PyTorch. Survival analysis is handled by incorporating a customizable loss function that implements the negative partial log-likelihood of the Cox model including both tunable L1 and L2 regularization. Moreover, DeepHazard provides optimized support for transfer learning applications, enhancing its applicability to small datasets. The package was developed and tested for precision medicine applications, but there are no conceptual limitations in its applicability across different domains.

The remainder of the paper is organized as follows: section 2 provides a theoretical background on the Cox model from which the implemented loss-function is derived; section 3 provides details about the implementation and the functionalities of the package; section 4 provides the references to download and use the package; finally, section 5 summarizes the conclusion of the paper.

2. Theoretical background

A detailed dissertation about all the possible approaches to survival analysis goes beyond the scope of this paper and can be found elsewhere [1]. However, here follows a few basic notion that are relevant for the reader the seeks to understand the presented implementation.

2.1. Target representation in survival analysis

Unlike traditional supervised learning tasks where labels are either discrete, such as the case of classification, or continuous, in the case of regression, survival analysis involves a more complex target representation, that is relevant to know when approaching the code of the presented package. In fact, each observation in a survival dataset consists of:

1. an event time T_i (a continuous numerical value): representing the time at which the event (or the last follow-up) occurs;
2. an event indicator δ_i (a binary value): representing a flag indicating whether the event was observed (1) or not observed (0).

For example, in a medical study tracking the survival of cancer patients, a patient's target might look like (24 months, 1) which means that after 24 months after the diagnosis the patient has experienced the event (*e.g.* death). Or it might look like (24 months, 0) which means that, up to the last available follow-up that occurs 24 months after the diagnosis, the patient has not yet experienced the event; in this case it is called *censored*.

2.2. Cox proportional hazard loss function

One of the milestones of survival analysis is the previously mentioned Cox proportional hazards model, that was introduced by Sir David Cox in 1972 [8]. The model is based on a semi-parametric approach that aims at estimating the hazard function, which quantifies the instantaneous risk of an event occurring at a specific time, given that the subject has survived up to that point. The hazard function is expressed as:

$$h(t|X) = h_0(t)e^{\beta X} \tag{1}$$

where:

- $h_0(t)$ is the baseline hazard function, representing the hazard when all covariates are zero; the model is considered semi-parametric because it makes no assumptions on the baseline hazard function;
- X is the vector of covariates (features);
- β is the vector of regression coefficients, which determine the effect of each covariate on the hazard.

The name *proportional hazards* refers to the fact that changes in the value of a predictor produce proportional changes in the hazard regardless of time.

Unlike standard regression models, the Cox model does not estimate explicitly the time to event. Instead, it focuses on risks and it estimates the coefficients by maximizing the partial likelihood function, which depends only on the ordering of event times rather than their exact values.

Given that the baseline hazard function is unknown, since no assumptions are made on its functional form, the standard likelihood cannot be used as a loss function, whereas it is often replaced in deep learning-based survival models by the negative partial log-likelihood function, given by:

$$L(\beta) = -\sum_i \delta_i (X_i\beta - log \sum_{j\ in\ R_i} e^{X_j\beta}) \tag{2}$$

where:

- δ_i is an event indicator, equal to 1 if the event occurred and 0 if the data is censored;
- R_i is the risk set, consisting of individuals who are still at risk at time t.

Such a loss function is designed to handle censored data, so that the model does not perform a standard regression aimed at estimating precise time-to-event, but rather it estimates the relative risk for each patient (or sample) and outputs a ranking, ordering patients according to their risk level.

3. Key features and implementation

As already stated in section 1.2, the present article aims at presenting a novel python package for the implementation of deep learning models devoted to survival analysis that extends the regularized Cox proportional hazard model, making it more adaptable and powerful for complex datasets. The package provides a versatile and user-friendly solution both compatible with Tensorflow/Keras and PyTorch, in which survival analysis is handled by incorporating a customizable loss function that implements the negative partial log-likelihood of the Cox model including both tunable L1 and L2 regularization. The next subsections provides an overview of the key features of the package.

3.1. Customizable model architecture

DeepHazard offers a high degree of flexibility when defining the neural network architecture, which makes it an ideal tool for a wide variety of problems. The user can specify the input dimension (*i.e.* the number of features), the layer sizes (*i.e.* the number of neurons in each hidden layer), and regularization terms (L1 or L2 penalties) with their parameters.

The model is built on top of the Sequential class of Keras and PyTorch respectively, which allow the creation of layers in a simple and modular way, and inherit all the customizable hyper-parameters. The basic unit of the deep network are Dense layers, that can be included by specifying the layer size. The activation function defaults to the Scaled Exponential Linear Unit (SELU) [25] that should limit the issue with vanishing gradients. Dropout layers can be included by specifying a layer size between 0 and 1, representing the dropout probability.

Such a flexible architecture ensures that the model can adapt to a wide range of datasets, from the most simples with just a few features to the highly complex and multi-dimensional.

3.2. Loss function with L1 and L2 regularization

A key feature that sets DeepHazard apart from traditional Cox models is its custom loss function, coxph_neg_log_part_like_l1_l2, which combines the standard Cox proportional hazards loss with L1 and L2 regularization. The regularization terms are pivotal in deep learning applications because, together with other techniques such as early stopping or dropout, they prevent the model from overfitting, that is the most common issue when training deep neural networks, especially with small or noisy datasets that are typical of biomedical applications. In particular, L1 regularization induce sparsity, pushing the model to focus only on the most relevant features, whereas L2 regularization prevents excessively large weights, further stabilizing the model.

Thus, the resulting loss function can be formalized as:

$$L(\beta) = -\sum_i \delta_i (X_i\beta - log \sum_{j\ in\ R_i} e^{X_j\beta}) + \lambda_1 \sum_k |\beta_k| + \lambda_2 \sum_k \beta_k^2 \quad (3)$$

where λ_1 and λ_2 are the coefficients modulating L1 and L2 regularization respectively.

Another important feature is that DeepHazard utilizes *masking* to handle censored data, significantly improving computation efficiency.

3.3. Model saving and loading

DeepHazard provides robust functionality for saving and loading models, which is crucial for maintaining reproducibility and efficiency in machine learning workflows.

The save and load methods use dill, a Python library for serializing Python objects. When saving a model, DeepHazard stores the architecture and hyper-parameters (input dimensions, layer sizes, regularization terms, optimizer type, and activation function) as well as the values of the weights.

This allows for seamless reloading and deployment of models at any point in time, without the need to retrain the network from scratch. Moreover, this functionality ensures that a trained model can be stored and reused in future experiments or in production systems.

3.4. Transfer learning

Another key feature of DeepHazard is that it provides built-in support for transfer learning applications. In particular, each layer is associated with a flag indicating whether its weights should be modified during model fitting. The flag can be set through the set_trainable_layers method, that allows the user to freeze the weights of some layers of the model, effectively turning them into non-trainable layers. By freezing the early layers (which capture lower-level features), the user can retrain only the last few layers, which capture higher-level abstractions. This is particularly useful when adapting a pre-trained model to a new dataset or task, significantly speeding up the training process and preventing overfitting. It also facilitates the reuse of existing models for related tasks, making the development pipeline more efficient.

3.5. Prediction and predict_proba

It is important to remember that the output of this kind of networks is a risk, *i.e.* a quantity inversely related to survival, that provide a ranking between samples in the test/validation set. However, there are cases in which it might be useful to categorize such risk, for example distinguishing between high-, intermediate- and low-risk patients.

To this end, the predict_proba method is designed to apply customizable thresholds to the model's predictions. Specifically, it uses lt (lower threshold) and ht (higher threshold) values to classify predictions into (up to) three categories:

- [1, 0, 0]: for values below the lower threshold (indicating low risk);
- [0, 0, 1]: for values above the higher threshold (indicating high risk);
- [0, 1, 0]: for values in between the two threshold (indicating intermediate risk).

This method provides a way to interpret model predictions in terms of probability classes, making it ideal for risk stratification tasks.

With the idea of minimizing the overhead on users' code, survival-related evaluation metrics (*e.g.*, C-index or Brier score) were intentionally excluded from the package, since these can be easily found in standard survival analysis libraries such as *scikit-survival*.

4. Code availability

The source code of DeepHazard, as well as exemplary usage, is available at *https://gitlabdigei.aizoon.it/aizoOn/DeepHazard/*.

5. Conclusion

In this article, we presented the key features of DeepHazard, a powerful, flexible and user-friendly package that extend the traditional Cox proportional hazard model and leverages the capabilities of cutting-edge frameworks like Tensorflow/Keras and PyTorch for the implementation of deep learning feed-forward neural networks for survival analysis. With its customizable architecture, regularization options, built-in transfer learning support, and efficient handling of survival analysis tasks, it provides an excellent tool for researchers and practitioners working in many fields, ranging from biomedical sciences to finance or any other domain where survival data are prevalent.

Acknowledgements

This work was supported by the Italian Ministry of Research, under the complementary actions to the NRRP "D34Health – Digital Driven Diagnostics, prognostics and therapeutics for sustainable Health care" Grant (# PNC0000001).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for grammar and spelling check, or paraphrasing/rewording. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Wang, Y. Li, and C.K. Reddy. "Machine learning for survival analysis: A survey." ACM Computing Surveys (CSUR) 51.6 (2019): 1-36. doi:10.1145/3214306.

[2] I.G. Duncan, Actuarial applications of survival analysis in healthcare, Ph.D. thesis, Heriot-Watt University, Edinburgh, UK, 2020.

[3] H. Li, Y. Ge, H. Zhu, H. Xiong, and H. Zhao. Prospecting the Career Development of Talents: A Survival Analysis Perspective. Proceedings of the 23rd ACM SIGKDD International Conference, ACM Press, Halifax, NS, Canada, 2017, pp. 917–925. doi:10.1145/3097983.3098107.

[4] Z. Yang, J. Kanniainen, T. Krogerus, and F. Emmert-Streib. "Prognostic modeling of predictive maintenance with survival analysis for mobile work equipment." Scientific Reports 12 (2022): 8529. doi:10.1038/s41598-022-12572-z.

[5] G. Beis, A. Iliopoulos, and I. Papasotiriou. "An Overview of Introductory and Advanced Survival Analysis Methods in Clinical Applications: Where Have we Come so far?" Anticancer Research 44.2 (2024): 471-487. doi:10.21873/anticanres.16835.

[6] D. B. Doroshow and J. H. Doroshow. "Genomics and the History of Precision Oncology." Surgical Oncology Clinics of North America 29.1 (2020): 35-49. doi:10.1016/j.soc.2019.08.003.

[7] O. Akintunde, T. Tucker, and V. J. Carabetta. "The evolution of next-generation sequencing technologies." In: High Throughput Gene Screening: Methods and Protocols. Springer US, New York, NY, 2024, pp. 3-29.

[8] D. R. Cox. "Regression models and life tables." Journal of the Royal Statistical Society, Series B 34.2 (1972): 187-202. doi: 10.1111/j.2517-6161.1972.tb00899.x.

[9] I. H. Sarker. "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions." SN Computer Science 2 (2021): 420. doi:10.1007/s42979-021-00815-1.

[10] J. Schmidhuber. "Deep learning in neural networks: An overview." Neural Networks 61 (2015): 85-117. doi:10.1016/j.neunet.2014.09.003.

[11] S. Cong and Y. Zhou. "A review of convolutional neural network architectures and their optimizations." Artif Intell Rev 56 (2023): 1905-1969. doi:10.1007/s10462-022-10213-5.

[12] Y. Yu, X. Si, C. Hu, and J. Zhang. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures." Neural Computation 31.7 (2019): 1235-1270. doi:10.1162/neco_a_01199.

[13] M. Gezimati and G. Singh. "Deep learning for multisource medical information processing." In: Data Fusion Techniques and Applications for Smart Healthcare. Academic Press, 2024, pp. 45-76.

[14] M. Gholizade, H. Soltanizadeh, M. Rahmanimanesh, et al. "A review of recent advances and strategies in transfer learning." International Journal of System Assurance Engineering and Management (2025). doi:10.1007/s13198-024-02684-2.

[15] S. Wiegrebe, P. Kopper, R. Sonabend, et al. "Deep learning for survival analysis: a review." Artificial Intelligence Review 57 (2024): 65. doi:10.1007/s10462-023-10681-3.

[16] J. L. Katzman, U. Shaham, A. Cloninger, et al. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network." BMC Medical Research Methodology 18 (2018): 24. doi:10.1186/s12874-018-0482-1.

[17] Theano Development Team. "Theano: A Python framework for fast computation of mathematical expressions." arXiv preprint arXiv:1605.02688 (2016).

[18] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sonderby, D. Nouri, et al. "Lasagne: First release." (2015). doi:10.5281/zenodo.27878.

[19] T. Ching, X. Zhu, and L. X. Garmire. “Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data.” PLoS Computational Biology 14.4 (2018): e1006076. doi:10.1371/journal.pcbi.1006076.

[20] D. Wang, Z. Jing, K. He, and L. X. Garmire. “Cox-nnet v2.0: improved neural-network-based survival prediction extended to large-scale EMR data.” Bioinformatics 37.17 (2021): 2772-2774. doi:10.1093/bioinformatics/btab046.

[21] 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 32.1 (2018). doi:10.1609/aaai.v32i1.11842.

[22] H. Kvamme, Ø. Borgan, and I. Scheel. “Time-to-event prediction with neural networks and cox regression.” arXiv preprint arXiv:1907.00825 (2019).

[23] M. Monod, P. Krusche, Q. Cao, B. Sahiner, N. Petrick, D. Ohlssen, and T. Coroller. “TorchSurv: A Lightweight Package for Deep Survival Analysis.” Journal of Open Source Software 9.104 (2024): 7341. doi:10.21105/joss.07341.

[24] C. Nagpal, W. Potosnak, and A. Dubrawski. “auton-survival: an Open-Source Package for Regression, Counterfactual Estimation, Evaluation and Phenotyping with Censored Time-to-Event Data.” arXiv preprint arXiv:2201.00000 (2022).

[25] G. Lambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. “Self-Normalizing Neural Networks.” In: Advances in Neural Information Processing Systems 30 (2017). arXiv:1706.02515.