DPWTE: A Deep Learning Approach to Time-to-Event Analysis using a Sparse Weibull Mixture Layer

Achraf Bennis, Sandrine Mouysset, Mathieu Serrurier

DPWTE : A Deep Learning Approach to Time-to-Event Analysis using a Sparse Weibull Mixture Layer

Achraf Bennis[1], Sandrine Mouysset[1], and Mathieu Serrurier[1]

[1]Université Toulouse III, Paul Sabatier - I.R.I.T

Abstract

In this paper, we propose a deep-network-based approach that leverages a finite mixture of Weibull distributions to address a key challenge in time-to-event modeling: it comes to a parametric estimation of survival time distribution from censored data. The proposed model predicts the parameters of the mixture of Weibull distributions with respect to input variables. In addition, given that we set beforehand the upper bound of the mixture size, the model finds the requisite number of Weibull distributions to combine in order to model the event time distribution as accurately as possible. For this purpose, we introduce the *Sparse Weibull Mixture* layer that selects indirectly, through its weights, the Weibull distributions composing the mixture, whose mixing parameters are significant. To stimulate this selection, we apply a sparse regularization on this layer by adding a penalty term to the loss function that takes into account both observed and censored observations. We validate our method on both simulated and real datasets, showing that the proposed methodology yields a performance improvement over the state-of-the-art models proposed in this work.

Keywords— Survival Analysis, Weibull Distribution, Deep Learning, Sparse Regularization.

1 Introduction

The time-to-event analysis is one of the most widely used in the statistical analysis field in many areas such as e-commerce, finance, marketing, telecommunication, health field (e.g. cardiovascular death, tumor recurrence), and predictive maintenance (e.g., failure times of power grids, mechanical systems or electronic devices). This branch of statistics concerns the prediction of when a future event will occur. In this paper, we consider the methodology for time-to-event prediction. The time-to-event analysis as a field has primarily focused

on interpretability, arguably, at the expense of predictive accuracy. This is eventually the reason why standard machine-learning-based classifiers are commonly used in real-world applications while it would be more useful to apply survival methodology. Certainly, some classifiers may have the best scores in terms of accuracy. However, these binary models can only provide predictions for a predetermined point in time. One loses the interpretability and flexibility which is guaranteed by the modeling of the event densities as a function of time. Moreover, in survival data, it is common that a part of a population in which the event is not observed within the relevant time period, and could potentially occur after this recorded time or removed from the study, producing so-called *censored* data. In this case, the individuals of this sub-population provided us with censored times rather than event times. While this type of data is not taken into consideration by standard classifiers, the time-to-event analysis bridges this gap.

In our work, we propose an approach to time-to-event analysis: the event times distribution is assumed to follow a finite mixture of Weibull distributions, whose parameters depend on individual covariates. No particular assumptions about the nature of the relationship between the parameters and the features are made. It is up to the model, that we propose and which we call DPWTE, to estimate the optimal number of distributions it needs to combine for modeling the event times distribution. The parameter that represents this number is initialized to an upper bound before training the network. The novel approach described in this paper guarantees thus certain freedom of modeling with maximum precision. The main objective is therefore to estimate the optimal number of Weibull distributions that compose the mixture as well as their associated parameters. In short, DPWTE that stands for *Deep Parsimonious Weibull Time-to-Event* is a network-based model (Deep) with a special sparse layer (Parsimonious) that selects the optimal combination of Weibull distributions used for the time-to-event analysis. This paper makes the following contributions:

- We assume that the event times are drawn from a random variable that follows a finite mixture of Weibull distributions.
- We introduce a neural network-based model that finds the optimal combination (a.k.a mixture) of Weibull distributions (by estimating their parameters and mixing coefficients) to model a given event times distribution. This task is performed using a special layer called *Sparse Mixture Weibull*, whose purpose is to select, among all the estimated distributions, those that will be used in the final mixture. This process is stimulated by applying a sparse regularization on the weights given to each distribution by this layer. This means that we will add a penalty term in the loss function for this purpose.
- We consider the censored observations in the conception of the model.

The paper is organized as follows. Section 2 contains a summary of related work. In Section 3, we review some basic concepts from survival analysis. The section

4 is dedicated to a review of the mixture of Weibull distributions. In section 5 we describe the architecture of DeepWeiSurv that helps us to understand how DPWTE works. In Section 6 we describe the architecture of our proposed method for modeling a survival time distribution and highlight the role of the *Sparse Weibull Mixture* layer in this modeling. In Section 7, we conduct some simulations intending to verify that the method we propose behaves as expected. In Section 8 we evaluate our model on five real-world datasets and compare its performances with those of certain existing models, we add another experiment to assess the sensitivity of DPWTE to the censoring threshold. We conclude in Section 9.

2 Related Work

Kaplan-Meier estimator is considered as among the first estimators widely used for time-to-event prediction, but it doesn't incorporate individual covariates. In the time-to-event analysis that explores the relationship between features and both event and censored times, existing methods assume a linear dependence. The semi-parametric Cox Proportional Hazards [1] (CPH) model assumes the effect of covariates is a fixed and multiplicative covariate-dependent factor on the hazard rate (linear relationship) which may be too simplistic since, in the real-world data, the covariate effects are often non-monotonic. Thanks to the ability of neural networks to learn nonlinear functions, many researchers tried to model the relationship between the covariates and the time-to-event data. An extension of CPH with neural networks was first proposed by Faraggi and Simon[2] who replaced the linear risk of the Cox regression model, with one hidden layer multi-layer perceptron but without performance improvement. Katzman et al.[3] revisited the Cox model in the framework of deep learning (DeepSurv), which removes the proportionality constraint, and showed that it outperforms CPH in terms of C-index [4]. Cox-Time[5] which is also a Cox extension, does not require this assumption and uses an alternative loss function scaling well non-linear cases to remedy this constraint. Most previous works benchmark their methods against the random survival forests (RSF) [6]. RSF computes a random forest using the log-rank test as the splitting criterion. It computes the cumulative hazards of the leaf nodes and averages them over the ensemble. Hence, RSF is a very flexible continuous-time method that is not constrained by the proportionality assumption. Other previous works are based on Cox regression such as SurvivalNet[7], a network-based model using Bayesian optimization of the hyperparameters and Zhu et al.[8, 9] who proposed a convolutional neural network that replaces multi-layer perceptron architecture of DeepSurv and applied this methodology to pathological images. An alternative approach to time-to-event prediction is to discretize the duration and compute the hazard or survival function on this predetermined time grid. Lee et al. [10] proposed a method, called DeepHit, that estimates the probability distribution with a neural net and combines the log-likelihood with a ranking loss. Furthermore, the method has the added benefit of being applicable for competing

risks. Fotso [11] proposed N-MTLR that used a Multi-Task Regression (MTLR) as the base with a neural network that calculates the survival probabilities on the points of the time grid. Another interesting work proposed by Martinsson [12] in which he presented WTTE-RNN, a model for sequential prediction of time-to-event for censored data whose main role is to estimate the distribution of time to the next event as having a discrete or continuous Weibull distribution with parameters being the output of a recurrent neural network.

Unlike these discrete-time models DPWTE and DeepWeiSurv [13] model a time-to-event distribution and thus a continuous survival function that enables to estimate of the survival probability at any survival time horizon.

3 Survival Analysis

In this section, we are going to review some basic concepts in survival analysis. The main goal is to model the event time distribution as a continuous function of time. The survival function is the most common function that models this distribution

$$S(t_h) = P(T > t_h) = 1 - \int_0^{t_h} f(u)du = 1 - F(t_h) \tag{1}$$

with $f(t)$ and $F(t)$ denoting the probability density function and the cumulative distribution function of an event time T. There is an alternative characterization of the event time distribution which is given by the hazard rate $h(t)$ defined as follows

$$h(t) = \frac{f(t)}{S(t)} = \lim_{dt \to 0} \frac{1}{dt} P(t \leq T < t + dt | T \geq t) \tag{2}$$

The hazard rate is defined as the event rate at time t knowing the probability of survival at time t or beyond. In the real-world data, the true event times are not provided for all individuals. That is, apart from the true event times for some individuals (called non-censored data) the time recorded can be the follow-up time which is not long enough for the event to happen, or the time at which the individual left the study before its completion. In these last two cases, we then observe a right-censored time.

Let $\mathbf{X} = \{(\mathbf{x_i}, t_i, \delta_i) | i \leq n\}$ be a set of observations with $\mathbf{x_i} \in \mathbb{R}^d$, the i^{th} observation of the baseline data (covariates), $t_i \in \mathbb{R}$ its observed time associated, and δ_i indicates if the i^{th} observation is censored ($\delta_i = 0$) or not ($\delta_i = 1$). The likelihood for survival times is given by

$$\mathcal{L} = \prod_{i=1}^{n} f(t_i|\mathbf{x_i})^{\delta_i} S(t_i|\mathbf{x_i})^{\delta_i - 1} \tag{3}$$

4 Mixture Weibull Distributions

In this work, we assume that the event time follows a finite mixture of two-parameter Weibull distributions with respect to $\mathbf{x_i}$. In this case, we know the

formulas of S and h which are functions of Weibull parameters. This means that we just need to estimate the parameters of this finite mixture to model the event times distributions.

4.1 Particular Case: Single Weibull

Here is a particular case where the event time follows a single two-parameter Weibull distribution $W(\beta, \eta)$ with β and η are respectively the *shape* and the *scale* parameters which are strictly positive (by definition). In this case, we can estimate these parameters by the maximization of the likelihood method:

$$\underset{\beta>0,\eta>0}{\text{argmax}}\{\mathcal{L}(\beta,\eta)|(t_i,\delta_i)_i\} = \prod_{i=1}^{n} f_{W(\beta,\eta)}(t_i|\mathbf{x_i})^{\delta_i}.S_{W(\beta,\eta)}(t_i|\mathbf{x_i})^{1-\delta_i} \tag{4}$$

Alternatively, we can solve this problem by maximizing the log-likelihood $\mathcal{LL}$ (since the *log* function is monotonic):

$$\underset{\beta>0,\eta>0}{\text{argmax}}\{\mathcal{LL}(\beta,\eta)|(t_i,\delta_i)_i\} = \sum_{i=1}^{n} \delta_i log\big[f_{W(\beta,\eta)}(t_i|\mathbf{x_i})\big] + (1-\delta_i) log\big[S_{W(\beta,\eta)}(t_i|\mathbf{x_i})\big]$$

where:

$$S_{\beta,\eta}(t) = exp[-(\frac{t}{\eta})^{\beta}],$$
$$f_{\beta,\eta}(t) = (\frac{\beta}{\eta})(\frac{t}{\eta})^{\beta-1} S_{\beta,\eta}(t)$$

4.2 General Case: Mixture of Weibull Distributions

For now, we suppose that the event time T follows a finite mixture of two-parameter Weibull distributions conditionally to the baseline data features. In this context, it is easy to calculate S and h of this mixture at any time instant t. As this latter totally depends on the mixture parameters, we only need to estimate each couple of parameters of Weibull distributions that compose this mixture as well as its mixing parameters which is therefore the main objective.

Let T follows $\mathcal{W}_p = \sum_{k=1}^{p} \alpha_k W(\beta_k, \eta_k)$ a mixture of p Weibull distributions with its mixing coefficients $\alpha = (\alpha_1,...,\alpha_p)$ ($\sum_k \alpha_k = 1,\ \alpha_k \geq 0$) with $\beta = (\beta_1, ..., \beta_p)$ and $\eta = (\eta_1, ..., \eta_p)$ are respectively the vectors of *shape* and *scale* parameters of $\mathcal{W}_p$. We point out that, in statistics, the density of the mixture is a combination of its distribution densities. In other words, we have:

$$f_{\mathcal{W}_p} = \sum_{k=1}^{p} \alpha_k f_{W(\beta_k,\eta_k)}. \tag{5}$$

By definition, the log-likelihood of $\mathcal{W}_p$, considering the censored data, can

be written as follows:

$$\mathcal{LL}(\beta,\eta,\alpha|(t_i,\delta_i)_i) = \overbrace{\sum_{i=1}^{n}\delta_i log\Big[\sum_{k=1}^{p}\alpha_k f_{W(\beta_k,\eta_k)}\big(t_i|\mathbf{x_i}\big)\Big]}^{\mathcal{LL}_{\delta=1}} + \underbrace{\sum_{i=1}^{n}(1-\delta_i)log\Big[\sum_{k=1}^{p}\alpha_k(1-F_{W(\beta_k,\eta_k)}(t_i|\mathbf{x_i}))\Big]}_{\mathcal{LL}_{\delta=0}} \tag{6}$$

Thus, we estimate $\mathcal{W}_p$ parameters (α,β,η) by solving the *Maximum Likelihood Estimation* problem defined by the following equation:

$$(\hat{\beta},\hat{\eta},\hat{\alpha}) = \underset{\beta,\eta,\alpha,\beta\geq 1}{\operatorname{argmax}} \mathcal{LL}\Big(\beta,\eta,\alpha|\big(t_i,\delta_i\big)_i\Big) \tag{7}$$

As we notice in the equation 7, we set a constraint linked to the shape parameter. In fact, by definition, β and η are strictly positive. However, to assure the convexity of the $\mathcal{LL}$, we need to consider that β is at least equal to 1.

Let μ_i the mean lifetime of the i^{th} individual. Given that the mean of a mixture μ is a weighted combination of the means of the distributions that compose this mixture (i.e., $\mu = \sum_{k=1}^{p}\alpha_k\mu_k$) and knowing the Weibull's mean expression, we have:

$$\mu_i = \sum_{k=1}^{p}\alpha_k\eta_{ik}\Gamma(1+\frac{1}{\beta_{ik}}) \tag{8}$$

where Γ is the gamma function, β_{ik} and η_{ik} are the i^{th} components of β_k and η_k respectively. μ_i can used as an estimate of the survival time of the individual i.

5 Neural Network for Time-to-Event modelling with Mixture Weibull Distributions

As we said in the previous section, we consider the dependence of $\mathcal{W}_p$ parameters (Weibull parameters and the mixing coefficients) on the covariates of the baseline data. We propose to use a neural network to model this relationship.

Before moving on to the novel approach proposed in this paper, we begin by describing the parametric model of DeepWeiSurv [13] whose role is to model the event times distribution by a finite mixture of Weibull distributions.

5.1 Description of DeepWeiSurv

We name $f_p = \{f_p^1,..,f_p^n\}$ the set of functions that models the relationship between the baseline data covariates $(\mathbf{x_i})_i$ and the mixture parameters:

$$\begin{array}{rccc} f_p^i & : & \mathbb{R}^d & \rightarrow & \mathbb{R}^{p\times 3} \\ & & \mathbf{x_i} & \mapsto & (\alpha^i,\beta^i,\eta^i) \end{array}$$

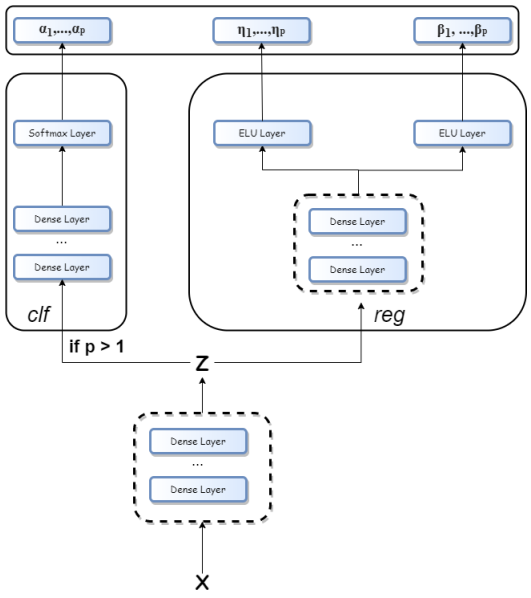


Figure 1: The architecture of DeepWeiSurv

where $\alpha^i = (\alpha_1^i, ..\alpha_p^i)$ are the mixing coefficients, $\beta^i = (\beta_1^i, .., \beta_p^i)$ and $\eta^i = (\eta_1^i, .., \eta_p^i)$ the parameters of p Weibull distributions that compose the mixture for the event of the i^{th}individual. To avoid being trapped by a notation problem, let's define α, β and η as follows:

$$\beta = (\beta_1, .., \beta_p) \quad = \begin{pmatrix} \beta_1^1 & \beta_2^1 & .. & \beta_p^1 \\ \beta_1^2 & .. & .. & .. \\ .. & .. & .. & .. \\ \beta_1^n & \beta_2^n & .. & \beta_p^n \end{pmatrix} \eta \quad = (\eta_1, .., \eta_p) = \begin{pmatrix} \eta_1^1 & \eta_2^1 & .. & \alpha_p^1 \\ \eta_1^2 & .. & .. & .. \\ .. & .. & .. & .. \\ \eta_1^n & \eta_2^n & .. & \eta_p^n \end{pmatrix}$$

and

$$\alpha = (\alpha_1, .., \alpha_p) = \begin{pmatrix} \alpha_1^1 & \alpha_2^1 & .. & \alpha_p^n \\ \alpha_1^2 & .. & .. & .. \\ .. & .. & .. & .. \\ \alpha_1^n & \alpha_2^n & .. & \alpha_p^n \end{pmatrix}$$

If $p = 1$, we don't need to estimate α since it is a vector of ones. The function f_p is represented by the network whose architecture is described in Figure 1 The role of this model is to learn the function f_p and thus estimate the mixture parameters (α, β, η). DeepWeiSurv is a multi-task network which consists of a common sub-network, a classification sub-network (denoted by *clf*) and a regression (denoted by *reg*). The shared sub-network takes as input the baseline data X and calculate a latent representation of the data denoted by Z. If the size of the mixture is greater than 1 ($p > 1$), then *clf* and *reg* sub-networks take the latent representation Z as an input and learn the final outputs which are the

parameters $\beta_1, .., \beta_p, \eta_1, .., \eta_p$ and the mixing coefficients $\alpha_1, .., \alpha_p$. We use ELU (by setting its constant to 1) as an activation function for both output layers of the *reg* sub-network. The advantage of ELU over the ReLU function is that we have enough gradient to learn the parameters thanks to its exponential form, it becomes thus smooth slowly. However the co-domain of ELU is $]-1, \infty[$. This is problematic since we have the constraints on the parameters β and η ($\beta \geq 1$, $\eta > 0$). To address this issue, the network will instead learn $\beta + 2 \geq 1$ and $\eta + 1 + \epsilon > 0$ where ϵ is a scalar strictly positive. We then apply the translation in the opposite direction to recover the parameters β and η. The *clf* sub-network, in turn, learn $\alpha \in \mathbb{R}^{n \times p}$. To ensure both conditions $\sum_{k=1}^{p} \alpha_k^i = 1, \forall i$ and $\alpha_k^i \in [0, 1], \forall (i, k)$, we use the *softmax* activation for *clf*'s output layer. Otherwise, i.e. $p = 1$, which represents the case of modelling by a single Weibull distributions where we don't need to learn α. Thus, we only need to train *clf* as described above.

In practice, in order to train the network-based model, we need to set a loss function. For DeepWeiSurv, we used the negative log-likelihood of the mixture. We will describe in detail the loss function used for DPWTE model in the next section.

6 Deep Parsimonious Weibull Time-to-Event Model (DPWTE): An extension of DeepWeiSurv

We remind that the goal is to model a time-to-event distribution with a finite mixture of Weibull distributions. The question that deserves to be asked is the number of Weibull distributions composing the mixture that is needed to model a given event time distribution. In other words, we need to know the estimated value of the mixture size p for the most accurate modeling of a specific survival time distribution. This problem was not identified in DeepWeiSurv model since p is fixed. For this model, we need to test different values of p to find the optimal one in terms of the model's performance. In this section, we will present and describe DPWTE model which proposes a solution to this issue.

6.1 Global Description of DPWTE Model

As discussed above, DeepWeiSurv fixes the value of p, the number of Weibull distributions used for distribution modelling. Hence, we don't know if this value is the optimal one, and this is represents one of the limitations of this model. DPWTE model addresses this issue, it has almost the same architecture as DeepWeiSurv, but the only difference is that, in the *clf* sub-network, we interleave a new layer called *Sparse Weibull Mixture* layer between the softmax layer and the output layer of *clf* as shown Figure 2 As for DeepWeiSurv, this network learns the triplet (α, β, η) that maximizes the likelihood of the mixture distribution. However, we will not necessarily use all the outputs to model the distribution, the parameter selection will be determined by the Sparse Weibull Mixture layer.

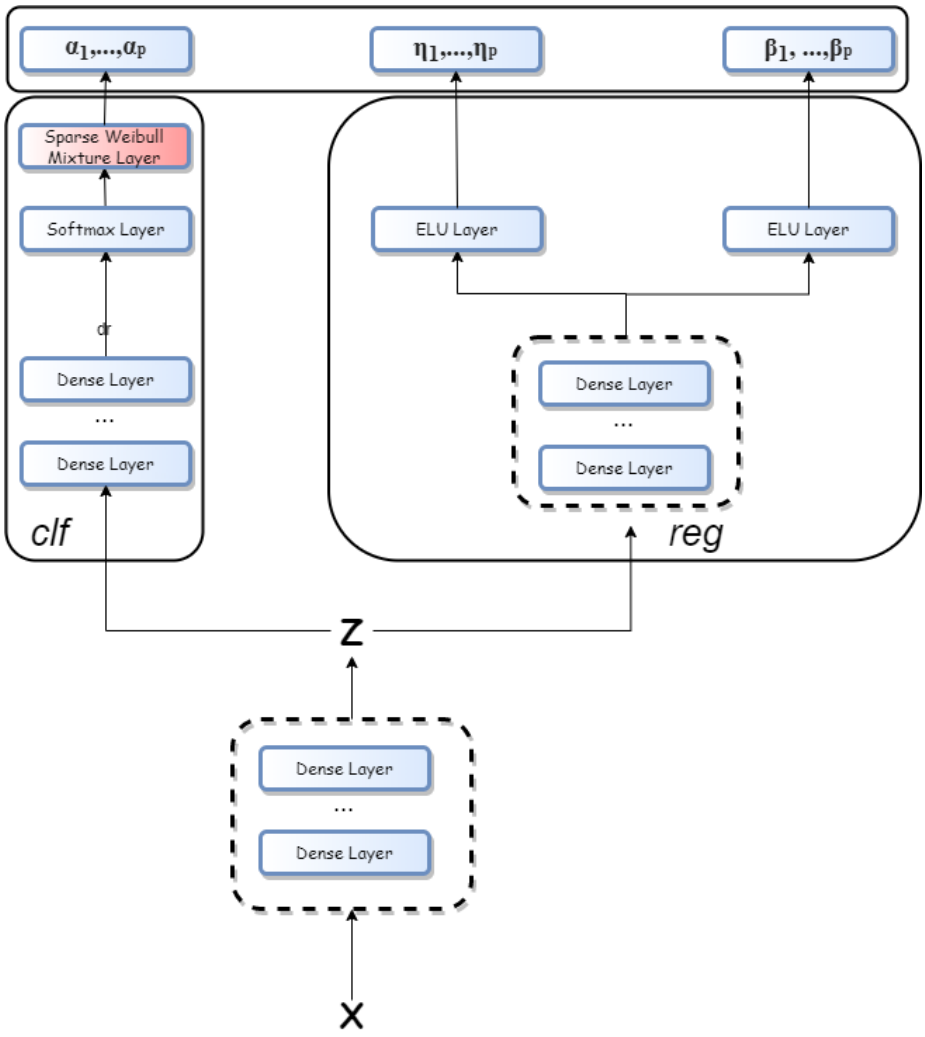


Figure 2: The architecture of DPWTE

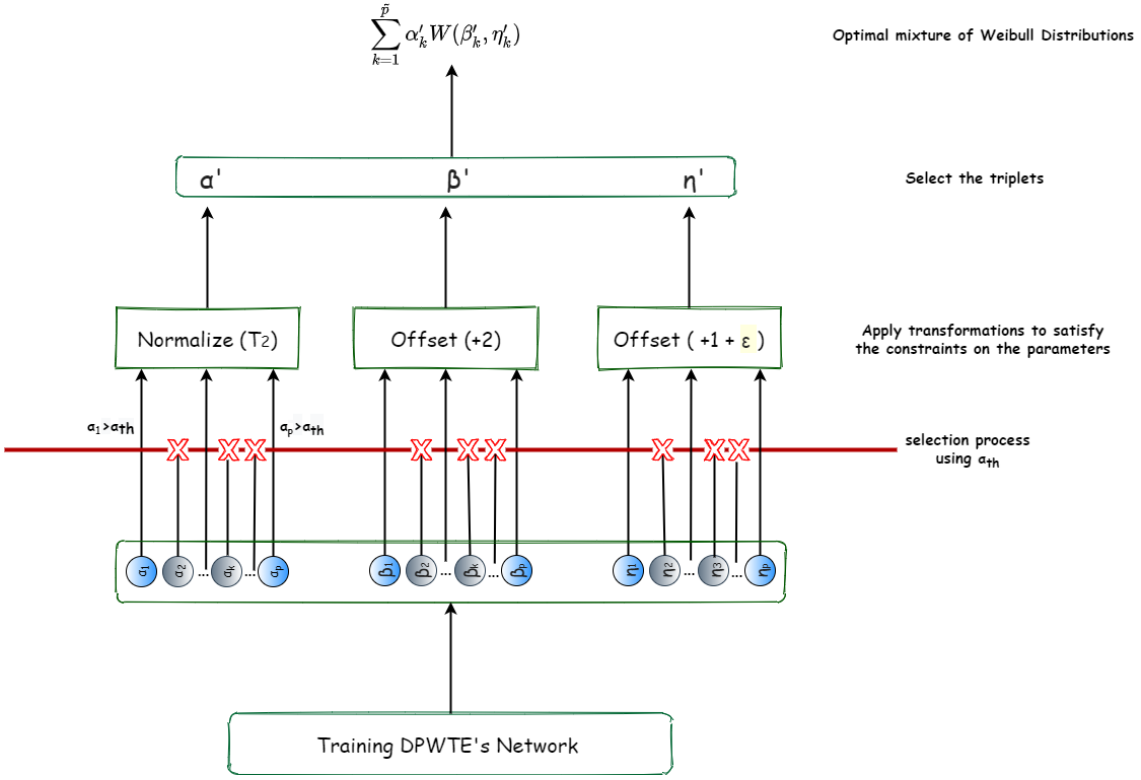


Figure 3: Schematic figure depicting the post-training steps of composing the optimal mixture of Weibull distributions.

6.2 Sparse Weibull Mixture Layer

It should be recalled that we try to find the number of Weibull distributions with which we compose the mixture to have the most accurate modeling of the event times distribution. In other words, we need to estimate p, the size of the mixture, denoted by $\tilde{p}$ initially set to an upper bound. The latter is set, beforehand, at a sufficiently large value. For this purpose, we introduced the Sparse Weibull Mixture layer. This layer performs an element-wise multiplication of the softmax layer in the *clf* sub-network. As we see in Figure 4, we have $\alpha_k = \omega_k * q_k$, where q_ks are the outputs of the softmax layer (in the case of DeepWeiSurv, $\tilde{p} = p$ and $q_k = \alpha_k, k = 1, .., p$). ω_k, through the importance of its value, can tell us if $W(\beta_k, \eta_k)$ should be used in the mixture. In order to interpret the value of ω_k to know which Weibull distributions must be used for the modelling, we need to satisfy the following conditions:

1. $\omega_k \in [0, 1]$
2. $\alpha_k \in [0, 1], k = 1, .., p$

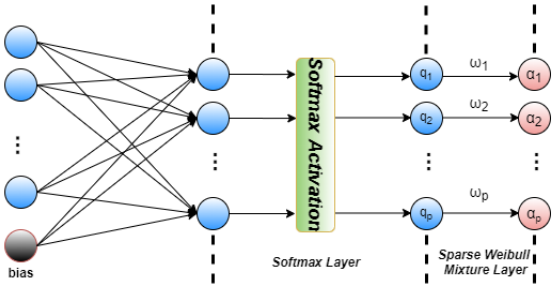


Figure 4: Softmax and Sparse Weibull Mixture layers of *clf*

3. $\sum_{k=1}^{p} \alpha_k = 1$

The constraint on ω_k (1) can't be guaranteed even if we initialize them in that way. The constraints (2,3) are established to guarantee that α is a probability vector. As $\alpha_k = \omega_k.q_k$ and $q_k \in [0,1]$, we have (1) $\implies$ (2) and thus the latter is automatically checked if (1) is checked. To ensure implicitly these constraints, we apply the following transformations:

$$(T_1)\ \ \omega_k \leftarrow \frac{|\omega_k|}{\sum\limits_{l}|\omega_l|}, k = 1, .., p \qquad (T_2)\ \ \alpha_k \leftarrow \frac{\alpha_k}{\sum\limits_{k}\alpha_k}, k = 1, .., p$$

6.3 Post-Training Steps: Selection of Weibull Distributions to Combine for Time-to-Event Modelling

So far, we have not yet estimated the value of $\tilde{p}$. The learning phase is the same as for DeepWeiSurv (even if they don't have the same loss function, we will see this in detail in section 6.4). However, after the DPWTE's network is trained, we select the triplets $(\alpha_k, \beta_k, \eta_k)$ such as α_k is greater or equal a certain threshold denoted by α_{th} that we fix beforehand. As we change the distribution of α after this selection (before training: $\alpha = (\alpha_1, .., \alpha_p)$ and after training and selection process: $\alpha = (\alpha_k, \alpha_k \geq \alpha_{th})$ but we want to keep the probability constraint, we thus need to apply the transformation (T2) to the new α. Therefore if $A = \{(\alpha_k, \beta_k, \eta_k) | \alpha_k \geq \alpha_{th}\}$ is the set of selected triplets for modelling, then:

1. $\tilde{p} = Card(A)$
2. $\alpha = (\alpha_k, \alpha_k \geq \alpha_{th}) \underset{T_2}{\longrightarrow} \alpha'$
3. $\beta = (\beta_k, \alpha_k \geq \alpha_{th}) \underset{offset(+2)}{\longrightarrow} \beta'$
4. $\eta = (\eta_k, \alpha_k \geq \alpha_{th}) \underset{offset(+1+\epsilon)}{\longrightarrow} \eta'$
5. the event times distribution can be modeled by $\sum\limits_{(\alpha_k, \beta_k, \eta_k) \in A} \alpha'_k W(\beta'_k, \eta'_k)$

This post-processing is described by the figure 3.

6.4 Loss Function

As discussed in previous sections, DPWTE is supposed to find the optimal combination of Weibull distributions. For these reasons, we propose the following loss function:

$$loss = -\mathcal{LL}(\beta, \eta, \alpha|(t_i, \delta_i)_i) + \lambda||\omega||_{\frac{1}{2}}, \tag{9}$$

where λ is the regularization parameter and $||\omega||_{\frac{1}{2}} = \sum_{k=1}^{p} \sqrt{|w_k|}$. The first element of the loss is the negative log likelihood (see Equation 6) which is used as a loss function for DeepWeiSurv. To stimulate the triplet selection process discussed in the previous section, we apply a *sparse* regularization on $\omega = (\omega_k)_{1 \geq k \geq p}$ by adding a penalty term (second operand) to the loss function, hence the name of Sparse Weibull Mixture layer and the word 'Parsimonious' in the name of the model. The purpose behind the sparse regularization is to encourage sparsity in the vector ω or at least some ω_k to become almost nil, and then apply the threshold α_{th}.

Clearly, the L_0 regularizer is ideal for this purpose in the sense of yielding the most sparse weights. However, the L_0 norm is non-differentiable, we cannot, therefore, incorporate it directly as a regularization term in the loss function. C. Louizos et al. [14] proposed a solution through the inclusion of a collection of non-negative stochastic gates, which collectively determine which weights to set to zero but it is a complex optimization problem that is difficult to be solved. The solutions of the L_2 regularizer are smooth, but they do not possess the sparse property. While the L_1 regularizer leads to a convex optimization problem, but it does not yield a sufficiently sparse solution. X. ZongBen et al. [15] proposed $L_{\frac{1}{2}}$ as the new regularizer which is more sparse than the L_1 regularizer while it is still easier to be solved than the L_0 regularizer. The sparsity property of $L_{\frac{1}{2}}$ was demonstrated by Fan et al. [16]. In this paper, we opt for $L_{\frac{1}{2}}$-regularization as we see in the *loss* function.

7 Experiments on Simulated Data

In this section, we conduct three experiments on synthetic datasets where the main goal is to empirically investigate our proposed approach. These simulations are by no means exhaustive but are intended to verify that the methodology behind behaves as expected. In the first experiment, we evaluate the ability of DPWTE to learn the relationship between the mixture parameters and the baseline data features. Three scenarios are simulated with different functions of different levels of complexity as well as different mixture sizes. In the second simulation, we perform a clustering based on the weighting coefficient to evaluate the ability of DPWTE in estimating the weighting coefficients α and the size of the mixture needed to do so. We generate different shapes of datasets namely noisy moons, noisy circles, and noisy blobs. In the third and last experiment,

we evaluate the ability of the network in handling different levels of censoring settings. We test three different shapes of mixture: uni-modal (i,e. with one peak), bi-modal, and tri-modal mixtures. We would point out that these experiments were also run with different values of α_{th} $(10^k, k = -2, -3, -4, -5)$ to test different levels of tolerance and check whether the respective results are better. However, the same patterns were found in these settings, we therefore only keep the following value of the threshold $\alpha_{th} = 0.1$ in this paper. We would also highlight that $\tilde{p}$ should not always equal the number of Weibull distributions with which we simulate survival times (see an example in Section 7.4).

7.1 Network Configuration

DPWTE consists of a shared sub-network which is a 4-dense-layer network where the batch normalization is applied immediately following the first fully connected layer. These four hidden layers have 128, 64, 32, and 16 nodes respectively. The regression sub-network has two dense layers with 16 and 8 nodes respectively, whose the second one is batch normalized and two ELU output layers, while the classifier sub-network is composed of 2-dense layers with 16 and 8 nodes respectively, a softmax layer followed by the Sparse Weibull Mixture layer whose weights are initially generated using the uniform distribution of support [0,1]. The hidden layers are activated using ReLU function. The network is trained via Adam optimizer with a learning rate of 10^{-4}. We initialize the mixture size with $p_{max} = 10$, set the regularization parameter $\lambda = 10^{-4}$ and finally set the threshold $\alpha_{th} = 0.1$ for the post-training operations.

7.2 Experiment I: Parameters-Features Relationship Modelling

The purpose of this experiment is to investigate and evaluate DPWTE's ability to model the relationship between the baseline data features and the mixture parameters. Let $\mathbf{X}$ be a vector, of size $n = 5000$, of one-dimensional samples drawn from the uniform distribution $\mathcal{U}_{[0,1]}$ of support [0,1]. We consider three scenarios where we seek to reproduce the relationship between the covariates $\mathbf{X}$ and the mixture parameters with which we draw survival time samples, used to train DPWTE. We assume for this experiment that all the samples are non-censored. The scenarios considered here are defined as follows:

- The survival times samples are drawn from a single Weibull distribution where the parameters are defined as follows:

$$\beta = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad\qquad \eta = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2}.$$

- We generate survival times from a single Weibull distribution with parameters defined by the following functions:

$$\beta = 2\sin\left(2\mathbf{X} + 1\right)\sin\left(1 + e^{\mathbf{X}}\right) + \frac{7}{2} \qquad \eta = \cos\left(1 + \mathbf{X}\right) e^{\mathbf{X}^2} + \frac{1}{2}.$$

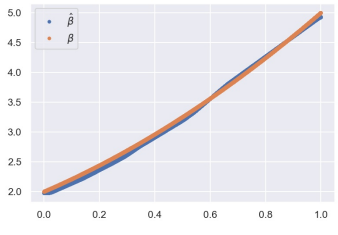
- We generate survival times from a 50-50 mixture of 2 Weibull distribution whose parameters $\beta_1, \eta_1, \beta_2, \eta_2$ are defined by the following functions:

$$\beta_1 = \mathbf{X} + 2 \qquad \eta_1 = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2}.$$
$$\beta_2 = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad \eta_2 = \frac{1}{2}(\mathbf{X} + 1).$$

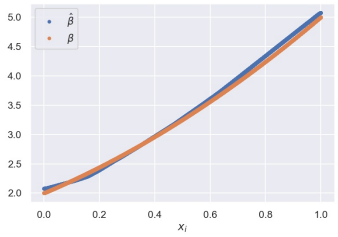
We compare the results of DPWTE and DeepWeiSurv. In the two first scenarios, we use DeepWeiSurv with $p = 1$, while in the last, we use DeepWeiSurv with $p = 2$. We set $p_{max} = 10$ for all scenarios. For each case study, we plot the simulated parameters β, η against their respective estimates. In these defined scenarios, we do not need to plot the estimated expectation since, using this data configuration, it has the same shape as the parameter η. The results of the first scenarios are displayed in Figure 5 and those of Scenario 3 in Figure 6.

For the first scenario, we train DPWTE and obtain the mean of $\hat{\alpha}$ over the 5000 instances: $(2.1e-5, 4.15e-7, 2.9e-4, 9.976e-1, 3.15e-8, 1.8e-3, 9.97e-10, 4.8e-7, 2.57e-4, 3.107e-10)$, where, by applying the threshold α_{th}, the fourth distribution (average of $\hat{\alpha}_4 = 0.9976$) is the only distribution selected and thus we obtain $\tilde{p} = 1$ which seems logical, since the survival times are drawn from one Weibull distribution. The results of the estimate parameters $\hat{\beta} = \hat{\beta}_4, \hat{\eta} = \hat{\eta}_4$ associated are plotted in Figure 5a and Figure 5c respectively, and those of DeepWeiSurv associated are shown in Figure 5b and 5d. We can notice that DPWTE and DeepWeiSurv have practically the same performance. In the second scenario, where the relationship between the Weibull parameters and $\mathbf{X}$ are more complex than that in the first case study for no other reason than that we have a composition and product of exponential and trigonometric functions, whereas in the first scenario, the relationship is polynomial of degree 2. The trained network outputs α whose mean is equal to ($1.0333e^{-5}$, $7.7868e^{-7}$, $5.8331e^{-6}$, $1.5778e^{-4}$, $3.5203e^{-7}$, $9.9988e^{-1}$, $5.4700e^{-6}$, $7.9631e^{-7}$, $6.0515e^{-7}$, $2.6980e^{-5}$), therefore by applying the threshold α_{th}, we obtain $\tilde{p} = 1$ and the post-training steps only select the sixth distribution since α_6 is the only weighting coefficient above the threshold. The results of the estimate parameters $\hat{\beta} = \hat{\beta}_6, \hat{\eta} = \hat{\eta}_6$ associated are plotted in Figure 5e and 5g respectively, and those of DeepWeiSurv associated are shown in Figure 5f and 5h. As we notice, non-smooth parts of the curve aside, both models provide a good approximation of η (Figure 5g) as for DeepWeiSurv (Figure 5h), while for β, they had more difficulty to model the function (Figure 5e and Figure 5f for DPWTE and DeepWeiSurv respectively).

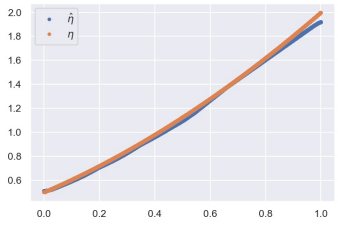
For the last case study, we train DPWTE and obtain the estimation of the weighting coefficients $\hat{\alpha}$ whose mean over the samples is (0.447, 0.46, $9.5e^{-4}$, $7.659e^{-4}$, $8e^{-4}$, 0.02, $5e^{-3}$, 0.065, $4.84e^{-4}$, $9.9e^{-8}$), which means that, after applying the threshold $\alpha_{th} = 0.1$, the process selects the two first distributions to model the simulated mixture, which means that $\tilde{p} = 2$. We notice that the average of $\hat{\alpha}_1$ and $\hat{\alpha}_2$ are close to each other and this implies that after normalizing, we obtain approximately a 50-50 mixture. Now let's describe the



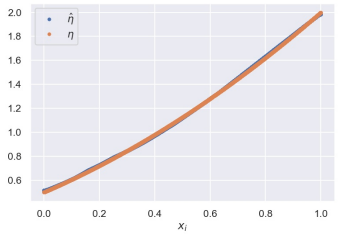
(a) DPWTE: β in Scenario 1



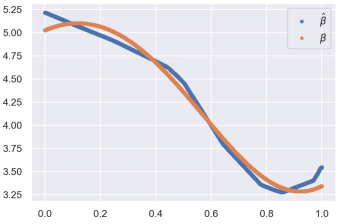
(b) DeepWeiSurv: β in Scenario 1



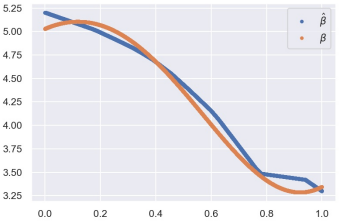
(c) DPWTE: η in Scenario 1



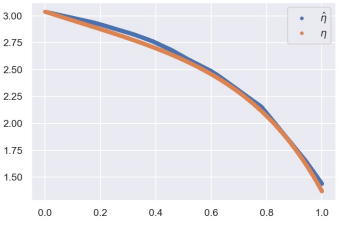
(d) DeepWeiSurv: η in Scenario 1



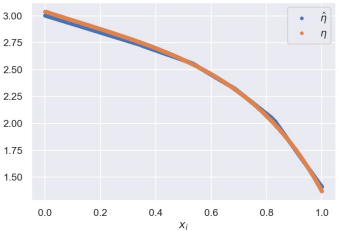
(e) DPWTE: β in Scenario 2



(f) DeepWeiSurv: β in Scenario 2

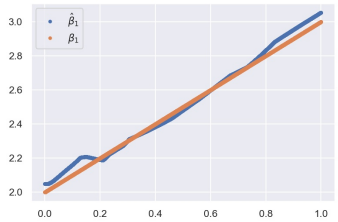


(g) DPWTE: η in Scenario 2

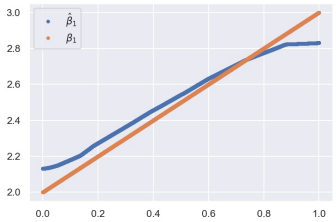


(h) DeepWeiSurv: η in Scenario 2

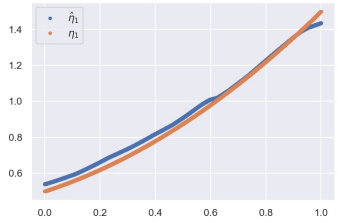
Figure 5: Results of DPWTE and DeepWeiSurv in Scenarios 1 and 2.



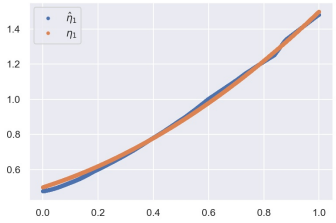
(a) DPWTE: $\hat{\beta}_1$ vs β_1



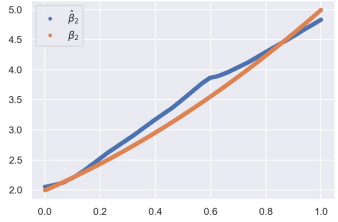
(b) DeepWeiSurv: $\hat{\beta}_1$ vs β_1



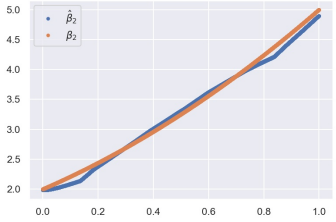
(c) DPWTE: $\hat{\eta}_1$ vs η_1



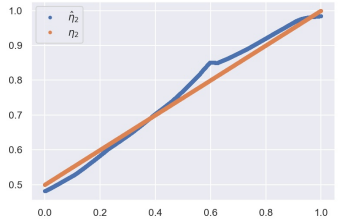
(d) DeepWeiSurv: $\hat{\eta}_1$ vs η_1



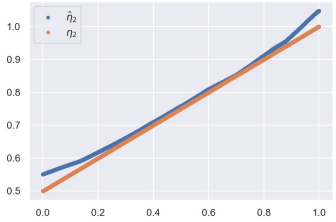
(e) DPWTE: $\hat{\beta}_2$ vs β_2



(f) DeepWeiSurv: $\hat{\beta}_2$ vs β_2



(g) DPWTE: $\hat{\eta}_2$ vs η_2



(h) DeepWeiSurv: $\hat{\eta}_2$ vs η_2

Figure 6: Results of DPWTE and DeepWeiSurv for Scenario 3.

results of the mixture parameters selected shown in Figure 6. In this figure, we notice that DeepWeiSurv provides estimate values that practically coincide with the real ones. The same goes for DPWTE which has the same performance as DeepWeiSurv but the only difference is that the model started by modeling the mixture with p_{max}=10 Weibull distributions and found out, after training, that he only needs 2 Weibull distributions to do so which coincide with the exact size of the simulated mixture. This means that DPWTE does not need to have prior knowledge about the size of the simulated mixture and this can be considered as an advantage over DeepWeiSurv in a real-world setting.

7.3 Experiment II: Clustering

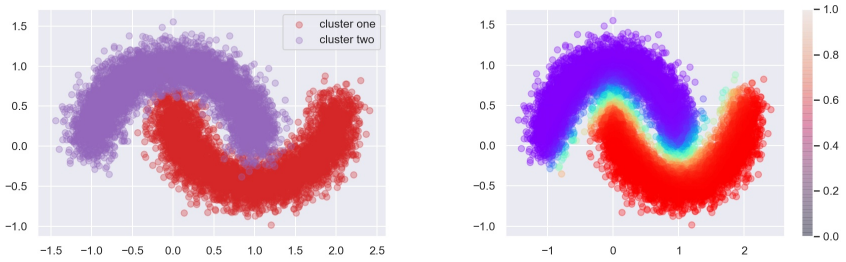
In this experiment, we want to evaluate the ability of DPWTE in estimating the weighting coefficients used to generate the simulated survival times. For this purpose, we consider the problem of clustering in this experiment. The main idea here is to generate m clusters $\mathcal{C}_1, .., \mathcal{C}_m$ not linearly separable. Each cluster $\mathcal{C}_i$ is labeled by a Weibull distribution. In other words, the samples that belong to a cluster $\mathcal{C}_i$ have their respective survival times that are drawn from the same single Weibull distribution of parameters β_i, η_i. We test three case studies:

- We draw 10000 samples using the function *make_moon* (Figure 7a) from the scikit-learn package of Python, half of which forms the first cluster $\mathcal{C}_1$ and the second half forms the second one $\mathcal{C}_2$. These clusters are in a shape of a moon in the waxing crescent phase hence the name of the function. We add a standard deviation of Gaussian noise to have thick clusters ($std = 0.15$). We draw for the samples of $\mathcal{C}_1$, time outputs from a single Weibull distribution whose parameters are $\beta_1 = 6.5, \eta_1 = 5$, whereas for the samples belonging to $\mathcal{C}_2$ have their survival times drawn from a Weibull distribution of parameters $\beta_2 = 1.5, \eta_2 = 0.5$.

- We draw 10000 samples using the function *make_circles* (Figure 8a) from scikit-learn package of Python, regularly distributed on two clusters $\mathcal{C}_1, \mathcal{C}_2$. These clusters are two nested circles, i.e. a large circle containing a smaller circle in 2d. We apply a standard deviation on 0.1 to generate noisy circles. For each sample i of the small cluster $\mathcal{C}_1$, we draw t_i the time output from a Weibull distribution $\beta_3 = 3.5, \eta_3 = 2.5$ and for each sample j from the bigger cluster $\mathcal{C}_2$, we draw t_j from $W(\beta_4 = 5, \eta_4 = 7.5)$.

- We draw 15000 samples using the function *make_blobs* (Figure 9a) from scikit-learn package to generate three isotropic clusters with a Gaussian noise (standard deviation of 0.8), fairly shared among the three clusters $\mathcal{C}_{i=1,2,3}$. $\mathcal{C}_1$ is characterized by $W(\beta_1, \eta_1)$, $\mathcal{C}_2$ is characterized by $W(\beta_2, \eta_2)$, while $\mathcal{C}_3$ is labeled by $W(\beta_3, \eta_3)$.

In each scenario, the Weibull distributions with we which we label the clusters are chosen in a such way that their respective densities are markedly separated in time. As in the previous experiment, we assume that all samples are non-censored. We recall that the objective of this experiment is to check if the model



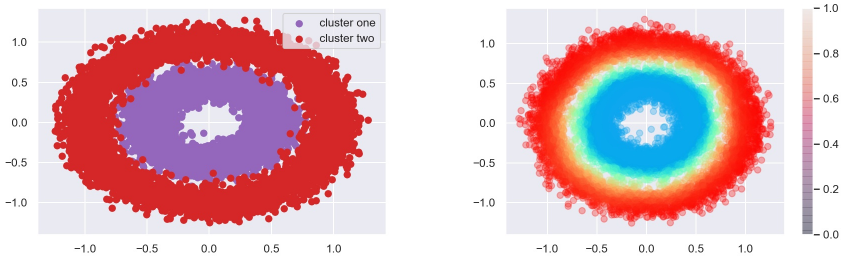
(a) The drawn samples (b) The samples coloured w.r.t $\hat{\alpha}_{i2}$

Figure 7: On the left, two noisy moons generated using the *make_moon* function from scikit-learn package. On the right, the clustering reproduced by DPWTE using the values of the second component of $\hat{\alpha}(\tilde{p} = 2)$.

can reproduce the clustering. We train DPWTE in each scenario, and obtain the estimate $\hat{\alpha}_i = (\hat{\alpha}_{i1}, .., \hat{\alpha}_{i\tilde{p}})$ for each sample $i \in \bigcup_{k\in[m]} \mathcal{C}_k$ after applying the threshold $\alpha_{th} = 0.1$ and α-normalization, where $\tilde{p}$ is the estimated size of the mixture predicted and $\hat{\alpha}_{ik} = \mathbb{P}(i \in \mathcal{C}_k)$ is the probability that the sample i belongs to the cluster $\mathcal{C}_k$. Since $\sum_{k=1}^{\tilde{p}} \hat{\alpha}_{ik} = 1$ as well as $\hat{\alpha}_{ij}$ and $1 - \sum_{k\neq j} \hat{\alpha}_{ik}$ are complementary, the latter thus provide the same information. We use the colormap to represent the intensity of the variable to visualize (between 0 and 1).

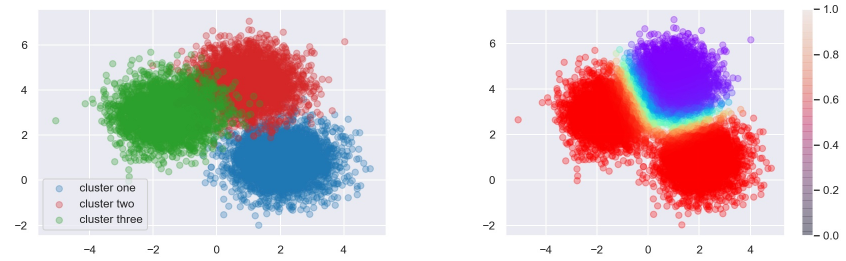
7.3.1 Results and Discussion

In the two first scenarios, we obtain $\tilde{p} = 2$, whereas, in the third scenario, we obtain $\tilde{p} = 3$, which means that DPWTE estimate exactly the optimal mixture size in these three cases. In Figure 7 and Figure 8 corresponding to Scenario 1 and 2 respectively, we visualize $(\hat{\alpha}_{i2}, i \in \mathcal{C}_1 \cup \mathcal{C}_2)$. We notice, for the moon-shaped clusters (7b), the big majority of samples from $\mathcal{C}_1$ and $\mathcal{C}_2$ have their value of $\hat{\alpha}_{i2}$ associated equal to 0 and 1 respectively. Only the samples that are in the edge of the cluster that faces towards the neighbor cluster have values between 0 and 1. Still, the border points in the edge of $\mathcal{C}_1$ are blue stained or colored in sky blue, which means that $\hat{\alpha}_{i2}$ converges to 0 (according to the colormap), and those of $\mathcal{C}_2$ is rather between yellow and orange, i.e $\hat{\alpha}_{i2}$ converges to 1. DPWTE performs a good clustering by starting with $p_{max} = 10$ Weibull distributions and finding the optimal mixture of size $\tilde{p} = 2$ as expected. For the nested-noisy-circle clusters, the samples of the outer noisy circle making $\mathcal{C}_2$ have almost all the value of $\hat{\alpha}_{i2}$ equal to 0.95, except those in the surface to the smaller circle whose $\hat{\alpha}_{i2}$ values are between 0.85 and 0.95 which still a good result (far from 0.5), whereas for the inner circle forming the cluster $\mathcal{C}_1$, the samples belonging

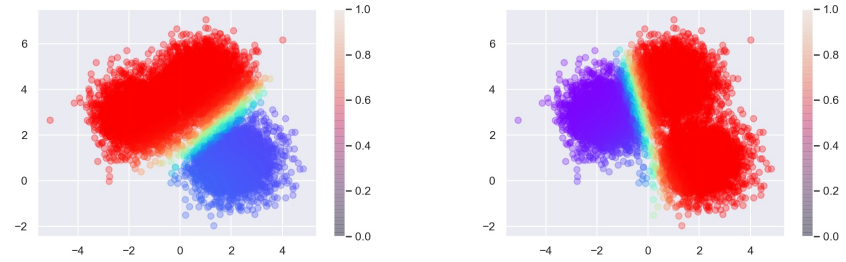


(a) The drawn samples. (b) The samples coloured w.r.t $\hat{\alpha}_{i2}$

Figure 8: On the left, two nested noisy circles generated using the *make_circle* function from scikit-learn package. On the right, the clustering reproduced by DPWTE using the values of the second component of $\hat{\alpha}$ ($\tilde{p} = 2$).



(a) The drawn samples. (b) The samples coloured w.r.t 1 - $\hat{\alpha}_{i2}$.



(c) The samples coloured w.r.t 1 - $\hat{\alpha}_{i1}$. (d) The samples coloured w.r.t 1 - $\hat{\alpha}_{i3}$.

Figure 9: Three clusters generated using the *make_blobs* function of scikit-learn package as well as their reproduced clustering performed by DPWTE ($\tilde{p} = 3$).

to the latter have their respective values of $\hat{\alpha}_{i2}$ not far from zero (colored in a blue sky) except for those that find themselves in the edge facing towards the outer noisy circle which has greater values but still acceptable (between blue sky and green, i.e. $\hat{\alpha}_{i2}$ way below 0.5).

In the third scenario, we have two degrees of freedom since $\hat{\alpha}_{i1}+\hat{\alpha}_{i2}+\hat{\alpha}_{i3} = 1$. We thus visualize for each cluster $\mathcal{C}_k$ the value of $1 - \alpha_{ik}$ (Figure 9c for $\mathcal{C}_1$, Figure 9b for $\mathcal{C}_2$ and finally Figure 9d for $\mathcal{C}_3$). We can notice that the majority of the samples belonging to the two last clusters are correctly labeled (colored in purple) ($1-\hat{\alpha}_{i2} = 0$ and $1-\hat{\alpha}_{i3} = 0$ respectively) except the border points on the edge facing toward the two neighbors where the values are not zero but still not far from zero. Whereas the non-border samples of the first cluster are blue coloured, which means that their respective values of $1 - \hat{\alpha}_{i1}$ are not exactly zero ($1-\hat{\alpha}_{i1}$ between 0.1 and 0.2) and the border points are blue sky coloured, still their respective values far below 0.5 which are acceptable values because by applying a threshold these samples are correctly labeled. We therefore conclude that DPWTE can have a good estimation of the weighting coefficients regardless the distribution of the baseline data and can also provide an improvement when compared to DeepWeiSurv (e.g. in the first scenario).

7.4 Experiment III: Censoring Threshold Sensitivity

The main objective in this experiment is to evaluate the performance of DPWTE with respect to the censoring rate, denoted by r_c, present in the data, i.e. the size of censored samples against the size of the data. We aim at each scenario defined by a value of r_c, reproduce the distribution simulated. Admittedly, the difficulty of modelling the distribution increases with the censoring rate, but also varies with the shape of the distribution. For this purpose, we run this experiment on three mixture distributions of different shapes:

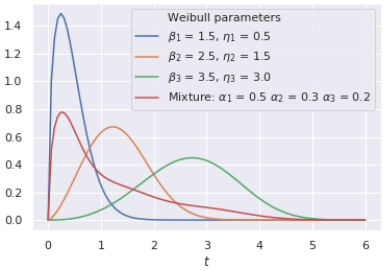
- *Uni-Modal Mixture*: we draw $m = 10000$ time samples from a 50-30-20 mixture of three Weibull distributions whose parameters are ($\beta_1 = 1.5, \eta_1 = 0.5$), ($\beta_2 = 2.5, \eta_2 = 1.5$) and ($\beta_3 = 3.5, \eta_3 = 3$) respectively with a weighting of 0.5, 0.3 and 0.2 respectively:

$$\mathcal{W}_3^1 = 0.5W(\beta_1, \eta_1) + 0.3W(\beta_2, \eta_2) + 0.2W(\beta_3, \eta_3).$$

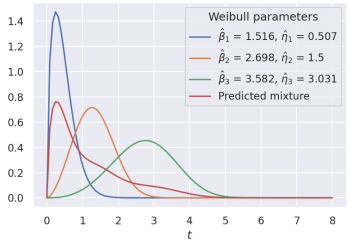
  The composing Weibull distributions and their mixture are illustrated in densities in Figure 10. For this mixture, we test the following values of censoring rate r_c: {0, 0.5, 0.7, 0.85, 0.95, 0.99}.

- *Bi-Modal Mixture*: we draw $m = 10000$ survival time samples from a 40-20-40 mixture of three Weibull distributions of parameters ($\beta_1 = 1.5, \eta_1 = 0.5$), ($\beta_2 = 2.5, \eta_2 = 1.5$) and ($\beta_3 = 4.5, \eta_3 = 3.5$) respectively with a weighting of 0.4, 0.2 and 0.4 respectively:

$$\mathcal{W}_3^2 = 0.4W(\beta_1, \eta_1) + 0.2W(\beta_2, \eta_2) + 0.4W(\beta_3, \eta_3).$$



(a) Simulated Weibull distributions $\mathcal{W}_3^1$



(b) Predicted Weibull distributions

Figure 10: The density of composing Weibull distributions simulated on the left, and those of predicted ones.

The composing Weibull distributions and their mixture are illustrated in densities in Figure 12a. For this mixture, we test the following scenarios of the value of censoring rate r_c: {0, 0.25, 0.45, 0.55, 0.65, 0.75, 0.85}.

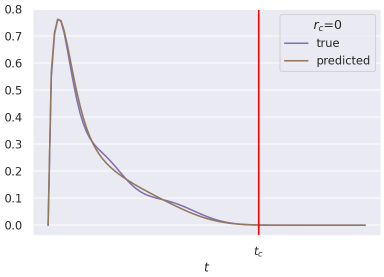
- *Tri-Modal Mixture*: we draw $m = 10000$ samples from a 40-30-30 mixture of three Weibull distributions of parameters $(\beta_1 = 1.5, \eta_1 = 0.5)$, $(\beta_2 = 5.5, \eta_2 = 1.5)$ and $(\beta_3 = 3.5, \eta_3 = 3)$ respectively with a weighting of 0.4, 0.3 and 0.3 respectively:

$$\mathcal{W}_3^3 = 0.4W(\beta_1, \eta_1) + 0.3W(\beta_2, \eta_2) + 0.3W(\beta_3, \eta_3)$$

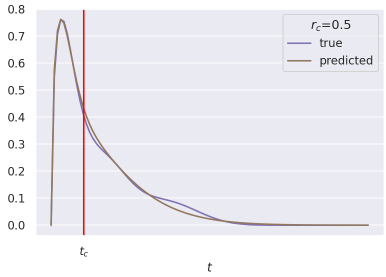
The composing Weibull distributions and their mixture are illustrated in densities in Figure 13. For this mixture, we test the following scenarios of the value of censoring rate r_c: {0, 0.1, 0.2, 0.3, 0.45, 0.55, 0.65, 0.85}.

7.4.1 Uni-Modal Mixture

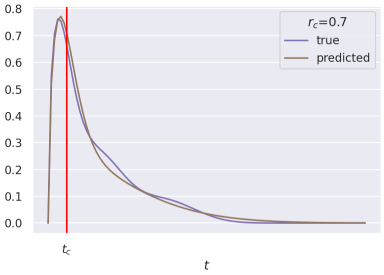
At an initial stage, we train DPWTE on the samples without considering the censoring rate, which means that we assume that all the time samples are non-censored, and obtain the predicted values of the triplets of parameters denoted by $(\hat{\alpha}_i, \hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$. We plot, as shown in Figure 10b, the predicted Weibull densities as well as the mixture of them using the parameters $(\hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$ and the predicted weighting coefficients $(\hat{\alpha}_i)_{i=1,2,3}$. As we can notice in Figure 10b and Figure 11a, the three distributions are reproduced and the predicted mixture coincides with the simulated one, which means that the mixture parameters and their weighting coefficients are correctly estimated by DPWTE. Now, let's see when we switch a portion of r_c of the data into a censored status. This means that the samples are split into non-censored sub-population (of size $\lfloor (1 - r_c) \times m \rfloor$) and censored sub-population (of size $\lfloor r_c \times m \rfloor + 1$). Then for each scenario defined by a value of the censoring rate, we train DPWTE on the resulting data and compare the mixture distribution with the one predicted by



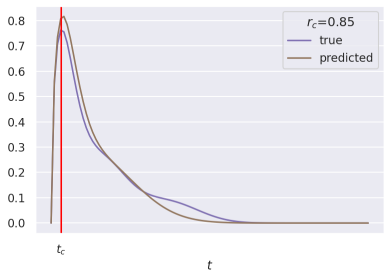
(a) $r_c = 0, \tilde{p} = 3$



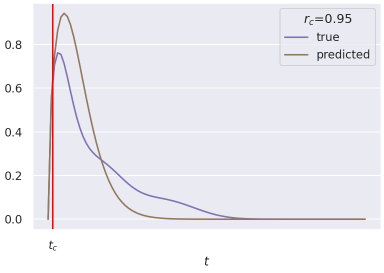
(b) $r_c = 0.5, \tilde{p} = 4$



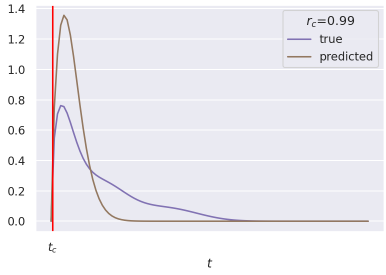
(c) $r_c = 0.7, \tilde{p} = 3$



(d) $r_c = 0.85, \tilde{p} = 2$



(e) $r_c = 0.95, \tilde{p} = 2$



(f) $r_c = 0.99, \tilde{p} = 1$

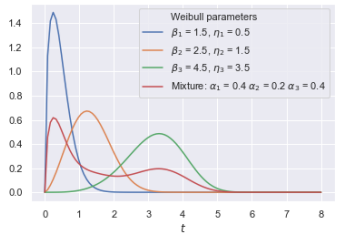
Figure 11: Results of the conducted experiment on the uni-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

the model, since the goal is to estimate the underlying distribution. We obtain the results for all the values of r_c tested in Figure 11. The purple curve corresponds to the probability density function of the mixture simulated while the brown curve is that of the predicted mixture. We recall that t_c corresponds to the censoring time, defined as the $(1 - r_c)$-th quantile of the vector of simulated times, above which a recorded time is considered as censored. This censoring time r_c is represented in the plots by a red vertical line.

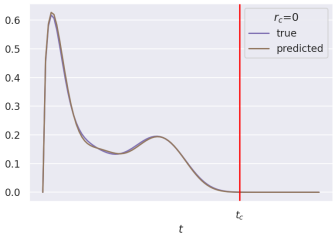
The first thing to notice is that we do not have the same value of the estimate $\tilde{p}$ from a scenario to another. This may be because sometimes, the network apportions a Weibull distribution to more than one node or because another Weibull distribution resembles that of the mixture with a weak weighting coefficient but shortly above the threshold α_{th}. We also suspect the decreasing of $\tilde{p}$ (while the censoring rate increases) comes from the fact that the more we increase the censoring rate the more the network ignores a part of a mixture and thus model the mixture with less Weibull distributions than it should be. We will see this example more clearly in the next scenario (bi-modal mixture). Another interesting thing that we can notice is that the precision of prediction decreases as the censoring rate increases which is expected since the increase of the censoring rate implies a loss of information about the overall distribution. Still, the model learns the shape of the distribution regardless of the value of r_c. As we noticed before, when $r_c = 0$, the two curves match up perfectly. For $r_c = 0.5$ and $r_c = 0.7$, the two densities almost coincide with each other. The same goes for $r_c = 0.85$ even the model does not have information about the tail of the distribution, but it predicts earlier density mitigation. For the extremely highly censoring setting namely the cases $r_c = 0.95$ and $r_c = 0.99$ (see Figure 11e and Figure 11f), the peak is overestimated which implies an early density attenuation. Still, the peak is well located even if it is not observed (the red vertical line is placed before the peak in these two cases). We can conclude that DPWTE can provide promising results in terms of handling the highly censoring setting as shown in this experiment.

7.4.2 Bi-Modal Mixture

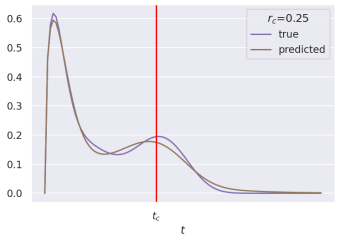
We conduct a similar experiment on the bi-modal mixture $\mathcal{W}_3^2$ described above, but with different values of r_c. We notice from Figure 12b that the simulated and predicted densities coincide with each other when $r_c = 0$. In this scenario, the model combines the same number of Weibull distributions to model the simulated one ($\tilde{p} = 3$). In the case where we have 25% of censored samples, DPWTE estimates the density of the mixture using three Weibull distributions ($\tilde{p} = 3$) with a very slight degradation of the precision (the second peak is slightly underestimated and slightly shifted on the left), however, it learns the underlying shape of the distribution and the positions of the peaks. For $r_c = 0.45$, the model predicted only the first peak which seems logical as the second peak is way located after the censoring time t_c. The model learned the first two distributions (hence $\tilde{p} = 2$) but completely ignored the second peak. The third highest value of α before normalization was of the order of 0.06 (less than the



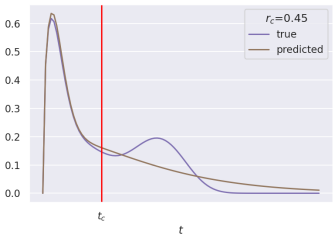
(a) Densities of $\mathcal{W}_3^2$.



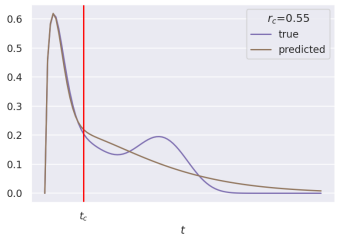
(b) $r_c = 0., \tilde{p} = 3$



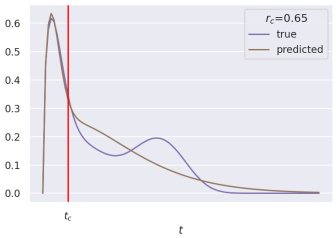
(c) $r_c = 0.25, \tilde{p} = 3$



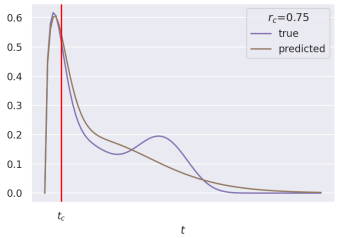
(d) $r_c = 0.45, \tilde{p} = 2$



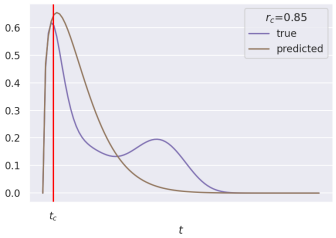
(e) $r_c = 0.55, \tilde{p} = 2$



(f) $r_c = 0.65, \tilde{p} = 2$



(g) $r_c = 0.75, \tilde{p} = 2$



(h) $r_c = 0.85, \tilde{p} = 1$

Figure 12: Results of the conducted experiment on the bi-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

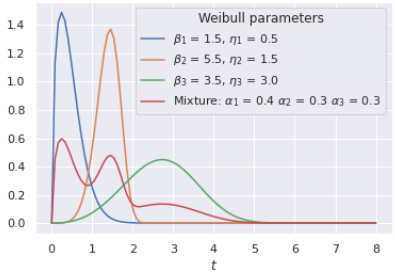
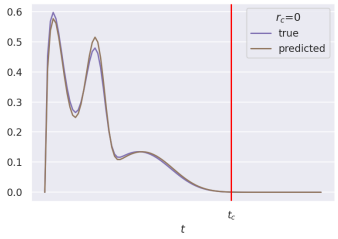


Figure 13: Densities of $\mathcal{W}_3^3$ and its composing distributions.

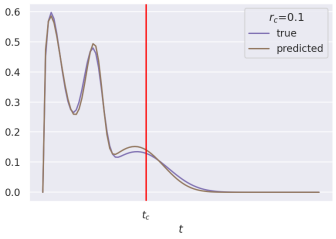
threshold 0.1), which corresponds to the third distribution that the model could not learn because of the high censoring. Still, the observed curve (before t_c) is perfectly estimated. The same goes for $r_c = 0.55$ and $r_c = 0.65$ where the model used 2 distributions, but it further ignores the tail of the distribution as the censoring time t_c further backward. For the last two cases, the respective predicted densities still ignoring the second peak while having earlier density mitigation. Compared to the uni-modal scenario, the degradation of the model performance (in terms of handling highly censoring setting) occurs before in this scenario. This is because, when the ratio of censored samples is important, the bi-modal mixture is considered more complex to learn than the uni-modal especially when the two peaks are largely separated in time.

7.4.3 Tri-Modal Mixture

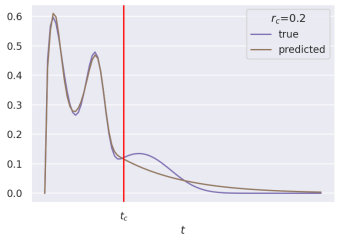
The same experimental protocol is executed on the tri-modal mixture $\mathcal{W}_3^3$ with different values of r_c. This means that the complexity of the problem grows even further. The third peak here appears but with a sharpness less important as seen in Figure 13. In the scenario where all the samples are non-censored, the model estimates with good precision the mixture with its three peaks correctly located. For $r_c = 0.1$, the model correctly predicted the peaks with their respective magnitudes and positions in time. For $r_c = 0.2$ and $r_c = 0.3$, the third peak starts to disappear, while the density values before this peak are perfectly estimated (the purple and brown curves coincide from t=0 to t=2 which corresponds to the interval that contains the first two peaks). For the cases $r_c = 0.45$ and $r_c = 0.55$, even if the second peak is not observed (it is placed after the red vertical line) the model predicts the latter with a very slight shift on the right but a significant overestimation of the magnitude which speeds up the density mitigation. For the last two cases, the precision of the density estimation continues to decline after the first peak, but still perfectly estimating the part of the curve corresponding to the observed times.



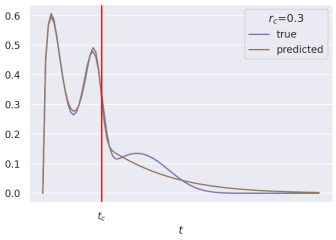
(a) $r_c = 0, \tilde{p} = 3$



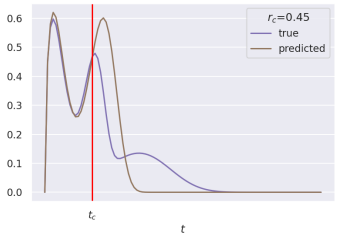
(b) $r_c = 0.1, \tilde{p} = 3$



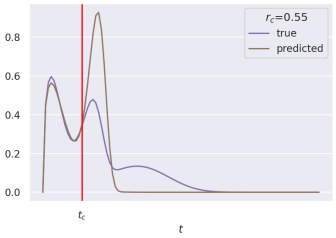
(c) $r_c = 0.2, \tilde{p} = 3$



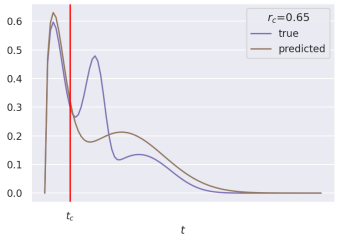
(d) $r_c = 0.3, \tilde{p} = 3$



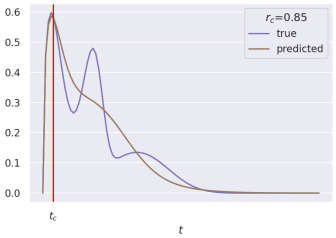
(e) $r_c = 0.45, \tilde{p} = 2$



(f) $r_c = 0.55, \tilde{p} = 2$



(g) $r_c = 0.65, \tilde{p} = 2$



(h) $r_c = 0.85, \tilde{p} = 2$

Figure 14: Results of the conducted experiment on the tri-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

7.4.4 Summary Results

To sum up, these three datasets represent different levels of difficulties in terms of modeling the mixture in a highly censoring setting. The difficulty of modeling in the presence of censored data depends on the number of peaks, their respective positions in time, and also their respective magnitudes. For example, let's take an example of two bi-modal Weibull distributions whose peaks have respectively the values (0.6,0.2) and (0.4,0.4). Since the magnitude of the second peak of the second Weibull is greater than that of the first Weibull. The latter is more likely to be ignored than that of the second Weibull because for a given value of r_c, the t_c associated is lower in the first case and thus the model further loses information.

In general, we can say that DPWTE handles censoring samples at a varying portion that depends on the shape of the distribution. For instance, in the uni-modal mixture, even with 99% of censoring, the model is considered to be well-performing until $r_c = 0.85$ but still learn the underlying shape of the distribution as well as the position of the peak, while in the bi-modal the deterioration in the quality of estimates starts earlier with notably $r_c = 0.45$. Finally, for the tri-modal mixture, the model has difficulties even earlier (it ignored the last peak from the case $r_c = 0.2$ and overestimated the magnitude of the second peak from the case $r_c = 0.45$ until it ignored it in the case $r_c = 0.85$), but in most cases, the position of the second peak is well predicted even if it is not observed.

8 Experiments on Real-World Datasets

,In this section, we evaluate our proposed model on real data sets and compare its predictive performance with the conventional benchmarks. We perform five sets of experiments: Breast Cancer, Heart Disease SEER, METABRIC, SUPPORT, and FLCHAIN. Table 1 gives an overview of descriptive statistics of these datasets. All the comparative models (including DPWTE) are tested in the same experimental protocol.

8.1 Description of the Real-World Datasets

The Surveillance, Epidemiology and End Results **SEER**[1] [17] Program provides cancer incidence data from population-based cancer registries covering approximately 34.6 percent of the U.S. population. For our experiment, we focused on the patients recorded between 1998 and 2002 with Breast Cancer (BC) or Heart Disease (HD) or who have survived to the end of this period. We kept 34 covariates including gender, race, tumor size, behavior code, Progesterone Receptor (PR) Status, Estrogen Receptor Status, etc. We yielded from this database two single-event datasets (BC and HD) keeping survivors in both of them.

[1] https://seer.cancer.gov

FLCHAIN: Assay Of Serum Free Light Chain is a database used to study the relationship between serum free light chain (FLC) and mortality. We used a stratified random sample containing 50% of the subjects from this database. We extracted 8 variables namely age, sex, kappa and lambda portion of serum-free light chain, FLC group, serum creatinine, and monoclonal gammopathy indicator MGUS.

SUPPORT is a sample of patients from a Study to Understand Prognoses Preferences Outcomes and Risks of Treatment. This dataset is very good for learning how to fit highly nonlinear predictor effects. We studied 9105 patients (of which almost 32% are survivors) with their 36 non-correlated attributes including age, sex, race, urine output creatinine, etc.

METABRIC: Molecular Taxonomy of Breast Cancer International Consortium is a north American dataset for a project of breast tumors classification. METABRIC contains gene expressions and clinical features including age, tumor size, PR Status, etc. In total, we have 1981 patients of which 888 died before the end of this study.

8.2 Network Configuration

DPWTE has a common sub-network which is a 2 fully connected layers (the batch normalization is applied before the second layer). The regression sub-network consists of 1 fully connected layer with batch normalization and two ELU layers as output layers, while the classifier sub-network is composed of 2 fully connected layers and a softmax layer followed by a *Mixed Weibull Sparse* layer. Hidden layers are activated by ReLU. The network is trained via SGD optimizer and learning rate of 10^{-4}.

8.3 Predictive Performance Metric

As evaluation metric, we use *concordance index* C-index [4] which calculates, among all the comparable pairs of observations (i, j) ($\delta_i = \delta_j = 1$), the number of concordant ones:

$$\text{C-index} = \frac{\sum_{i,j} \mathbb{1}_{t_i > t_j} . \mathbb{1}_{\hat{t}_i > \hat{t}_j} . \delta_j}{\sum_{i,j} \mathbb{1}_{t_i > t_j} . \delta_j}, \tag{10}$$

C-index estimates the probability of the event $\{\hat{t_i} > \hat{t_j} | t_i > t_j\}$ which compares the rankings of two independent and comparable pairs (non censored) of survival times (t_i, t_j) and the times predicted $(\hat{t_i}, \hat{t_j})$.

8.4 Experimental Setting

For evaluation, we applied 5-fold cross validation. For each iteration, the predicted parameters (α, β, η) and $\tilde{p}$ are used to calculate the mean lifetime μ and then the score C-index on validation set. The values of the scores displayed in Table 2 are the means of the scores calculated in each iteration. We set $p = 10$ and $\lambda = 10^{-4}$.

Table 1: Descriptive Statistics of Real-World Datasets

Datasets	No. Uncensored	No. censored	No. Features	Censoring Time			Event Time		
				min	max	mean	min	max	mean
SEER BC	9152(42.8%)	12221 (57.2%)	34	1	227	181.5	1	226	63.7
SEER HD	12014 (49.6%)	12221 (50.4%)					1	224	76.7
FLCHAIN	2169(27.6%)	5705(72.4%)	8	1	5215	4226.2	0	4998	2174.5
SUPPORT	5844(68.1%)	2735(31.9%)	36	344	2029	1060.2	3	1944	206.0
METABRIC	888 (44.8%)	1093 (55.2%)	21	1	308	116.0	1	299	77.8

8.5 Comparison Methods

For the five real-world datasets, which have a single event, the discriminative performance of **DPWTE** ($p = 10$) was compared with those of **DeepWeiSurv** with its parameter p set to 2, **CPH** with a penalty term = 0.1, Weibull Accelerated Failure Time model (**Weibull AFT**), Random Survival Forests (**RSF** [6]) with number of trees set to 100 and **DeepSurv** [3] with 2 layers of 32 nodes.

8.6 Results

We compare the tested methods using C-index that evaluates their predictive performances. Table 2 shows the concordance index averaged over the five cross-validation folds. We notice that, for the METABRIC experiment, DPWTE's prognostic performance exceeds by far that of the remaining models other than DeepWeiSurv (in the order of 0.16). For the SUPPORT dataset, DPWTE outperforms all the competing methods, other than DeepSurv and DeepWeiSurv, with statistically significant improvement and no overlap between DPWTE's performance and theirs in terms of the confidence interval. Whereas for the FLCHAIN dataset, DPWTE provides a slight improvement over all the comparison models other than DeepSurv, but still outperforms this latter on average. This might be because the FLCHAIN experiment represents a highly censoring setting (around 72% of censored observations) which is more challenging for the models. Regarding the SEER dataset, our model provides us a positive score differential, which is statistically significant, against all the competing models in the case of Breast Cancer. The same goes for the Heart Disease dataset excluding DeepWeiSurv which has a range of scores not significantly different from that of DPWTE. We can also remark that the standard deviation of the C-index in the METABRIC experiment is relatively greater than those of SEER, FLCHAIN, and SUPPORT datasets. We suspect that this comes from the fact that the METABRIC dataset size is relatively small in comparison with the other datasets. Another thing we would point out is that in all experiments other than the METABRIC case, DPWTE's confidence interval is narrower than those of the other methods other than DeepWeiSurv which means that DPWTE produces a more stable estimation. Therefore, as can be seen, our model consistently provides the best performance for the five datasets and shows a slight improvement over DeepWeiSurv. We suspect that the performance improvement comes from its capacity to find the optimal number of

Table 2: Comparison of C-index performance tested on Single Event Datasets (mean and 95% confidence interval)

Models	**Datasets**				
	SEER BC	SEER HD	FLCHAIN	SUPPORT	METABRIC
CPH	0.831 (0.824 - 0.839)	0.785 (0.781 - 0.788)	0.789 (0.783 - 0.794)	0.805 (0.799 - 0.813)	0.661 (0.635 - 0.687)
Weibull AFT	0.832 (0.825 - 0.839)	0.785 (0.78 - 0.79)	0.789 (0.784 - 0.795)	0.807 (0.802 - 0.814)	0.659 (0.634 - 0.684)
DeepSurv	0.841 (0.836 - 0.847)	0.786 (0.784 - 0.787)	0.79 (0.78 - 0.8)	0.826 (0.811 - 0.831)	0.662 (0.635 - 0.69)
RSF	0.838 (0.829 - 0.848)	0.755 (0.744 - 0.765)	0.75 (0.708 - 0.791)	0.783 (0.78- 0.789)	0.667 (0.636 - 0.699)
DeepWeiSurv	0.908 (0.906 - 0.909)	0.863 (0.86 - 0.868)	0.795 (0.79 - 0.797)	0.815 (0.79 - 0.82)	0.819 (0.812 - 0.837)
DPWTE	**0.912 (0.911 - 0.914)**	**0.871 (0.865 - 0.878)**	**0.802 (0.797 - 0.82)**	**0.83 (0.82 - 0.843)**	**0.829 (0.808 - 0.849)**

distributions needed to model the relationship between observations' covariates and the survival time distribution.

8.7 Censoring Threshold Sensitivity Experiment

In the previous experiment, the models learned on the original version of the considered benchmark datasets. This means that the censoring threshold is fixed in each dataset. In this experiment, for a given dataset $\mathcal{X}$, we train the models with different censoring thresholds that are greater than the initial one. In other words, let t_{c_0} denotes the initial censoring threshold for a given dataset $\mathcal{X}$, we choose k censoring thresholds $\{t_{c_i} | t_{c_i} > t_{c_0},\ i = 1, ..k\}$ where for each t_{c_i}, the original is transformed to a new set $\mathcal{X}_{c_i} = \{t_i \mid \delta(t_i) = 1 \text{ if } t_i < t_{c_i} \text{ and } 0 \text{ otherwise}\}$ with which the models are fitted. Then, for each censoring threshold t_{c_i}, we evaluate the associated models on all the $\mathcal{X}_{c_i}$. The goal here is to investigate the ability of the models to handle the highly censored settings. We conduct this experiment on FLCHAIN and METABRIC by testing the 4 best models that show a good performance in the previous experiment namely: DeepSurv, DeepHit, DeepWeiSurv with $p = 10$, and finally DPWTE. We choose METABRIC because of its size which is relatively small compared to the others adn thus renders the task more challenging, while FLCHAIN is selected because DeepHit has shown the best performance for this dataset and thus represents a fair choice.

8.7.1 Experimental Protocol

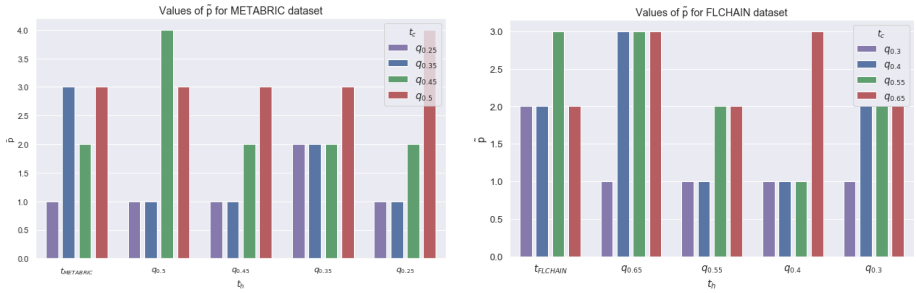
For each considered dataset in this experiment, the censoring thresholds are expressed in quantiles q_α of the event time variable and chosen in such a way that each threshold provides a significant portion of censoring compared to the one that precedes it, or in other words, changes significantly the time distribution. For METABRIC and FLCHAIN, we respectively choose the following censoring threshold vectors: $Q_{METABRIC} = (q_{0.5}, q_{0.45}, q_{0.35}, q_{0.25})$ and $Q_{FLCHAIN} =$

Table 3: Distribution of METABRIC observations (censored/non-censored) for each censoring threshold in $Q_{METABRIC}$.

t_c	No. censored	No. non-censored	Added portion
$t_{METABRIC}$	1093	888	-
$q_{0.5}$	1285	696	17.6%
$q_{0.45}$	1411	570	29%
$q_{0.35}$	1559	422	42.6%
$q_{0.25}$	1670	311	52.8%

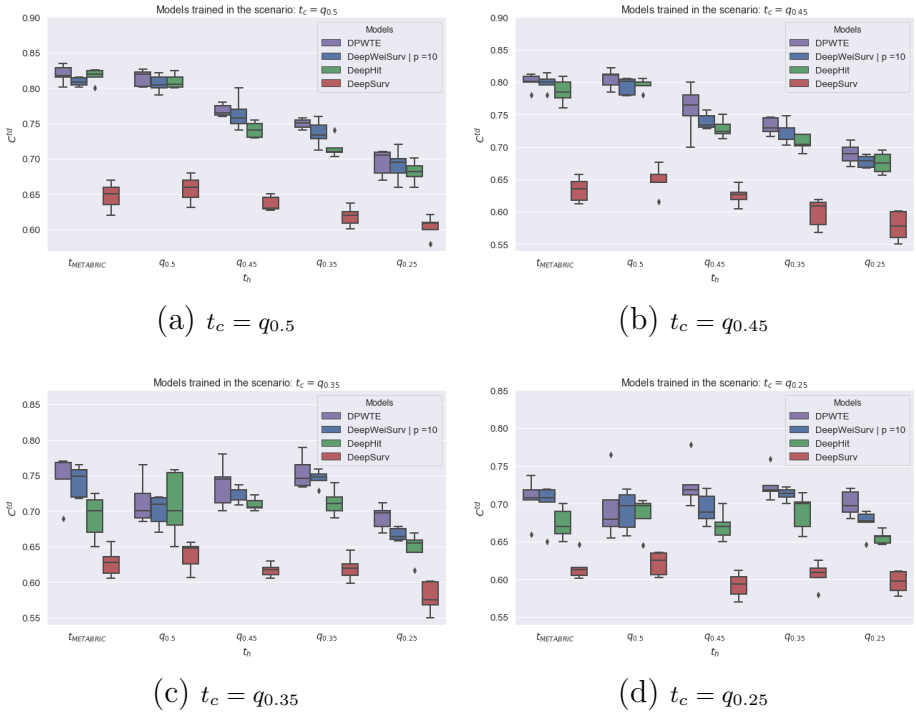
Table 4: Distribution of FLCHAIN observations (censored/non-censored) for each censoring threshold in $Q_{FLCHAIN}$.

t_c	No. censored	No. non-censored	Added portion
$t_{FLCHAIN}$	5705	2169	-
$q_{0.65}$	6237	1637	9.3%
$q_{0.55}$	6534	1340	14.5%
$q_{0.4}$	6820	1054	19.5%
$q_{0.3}$	7086	788	24.2%



(a) METABRIC (b) FLCHAIN

Figure 15: Mean values of the estimate $\tilde{p}$ calculated over the 5-fold cross validation for each censoring threshold t_c in both METABRIC (left) and FLCHAIN (right).



(a) $t_c = q_{0.5}$ (b) $t_c = q_{0.45}$

(c) $t_c = q_{0.35}$ (d) $t_c = q_{0.25}$

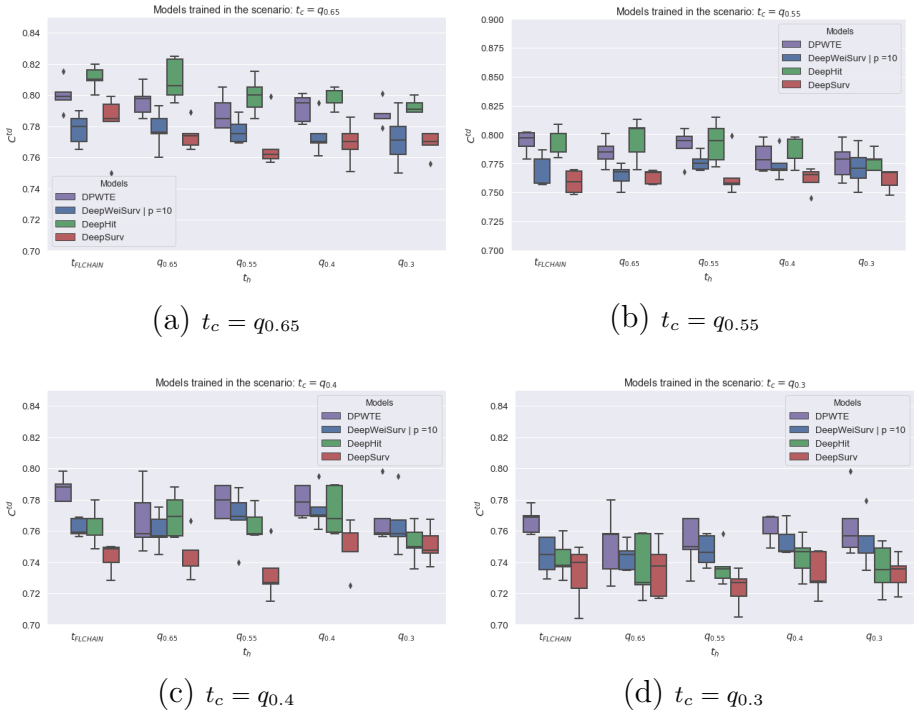
Figure 16: Box plots of the concordance index scores calculated, for each censoring threshold $t_c \in Q_{METABRIC}$ over the five-fold cross validation for METABRIC dataset.

$(q_{0.65}, q_{0.55}, q_{0.4}, q_{0.3})$. Table 3 and Table 4 give, for each censoring threshold, the associated distribution of censored/non-censored samples for METABRIC and FLCHAIN datasets respectively. The 'Added portion' column represents the percentage (out of the initial distribution) of data whose status switch from non-censored to censored.

For each scenario defined by a couple: censoring threshold t_{c_i} and associated dataset $\mathcal{X}_{c_i}$, we apply the five-fold cross-validation where, in each iteration, we train the models $\mathcal{M}$ on the associated training folds (4/5 of the global size) of $\mathcal{X}_{c_i}$ then evaluate these models on the associated validation fold of $(\mathcal{X}_{c_k})_{c_k \in Q_{\mathcal{X}}}$ by calculating the concordance index C^{td}. We obtain at the end of each scenario, a vector of five scores (calculated over the 5 iterations) for each model evaluation and each validation fold.

8.7.2 Results and Discussion

To highlight the mean and the standard deviation of the scores obtained, we use box plots. The results are shown in Figure 16 and Figure 17 for METABRIC and FLCHAIN datasets respectively where each scenario as described above is



(a) $t_c = q_{0.65}$ (b) $t_c = q_{0.55}$

(c) $t_c = q_{0.4}$ (d) $t_c = q_{0.3}$

Figure 17: Box plots of the concordance index scores calculated, for each censoring threshold $t_c \in Q_{FLCHAIN}$ over the five-fold cross validation for FLCHAIN dataset.

represented by a sub-figure. We call the time horizon t_h, the censoring threshold in the case of prediction. Firstly, it is considered overall, that the smaller the censoring threshold, i.e. the tighter the observation period, the weaker is the predictive performance of the models, which is (for the same reason as in the simulated experiments) normal since the quantity of knowledge about the distribution increases with the size of observed (non-censored) samples. For the METABRIC dataset, DPWTE outperforms in all scenarios whereas DeepSurv has the worst performance which means that it has the most difficulty in handling highly censored settings. As regards the two other models namely DeepHit and DeepWeiSurv, they still have a good performance with a normal drop for small thresholds. Still, the models generally perform an acceptable confidence interval. For FLCHAIN, DeepSurv gets closer to other models but still in the last of the ranking in terms of predictive performance. DeepHit starts (in the two first scenarios, i.e., for $t_c = q_{0.65}, q_{0.55}$) by outperforming the rest of models as in the previous experiments, but we notice that it has more difficulty than DPWTE and DeepWeiSurv for $t_c = q_{0.4}, q_{0.3}$. Furthermore, DeepHit performs standard deviations greater than those of DPWTE and DeepWeiSurv. We can

conclude that DPWTE and DeepWeiSurv are the best in handling highly censored settings with a slight improvement provided by DPWTE. We suspect that their performance comes from the fact that the Weibull distribution best fits the respective underlying distributions of the benchmark datasets. Another thing to notice, from Figure 15, is that the estimate $\tilde{p}$ globally decreases, regardless of the survival time horizon t_h, while decreasing the censoring threshold (hence the censoring rate is increasing). We suspect this comes from the fact that the more we increase the censoring rate the more the network ignores a part of the underlying distribution and thus model the latter with an insufficient combination of Weibull distributions (in terms of mixture size in this case).

9 Conclusion

In this paper, we proposed a novel approach for survival analysis. A network-based model, assuming a Weibull mixture character of the survival time, was presented to address this problem. We could, by parametrizing the mixture with neural networks, model rich relationships between the covariates and event times. DPWTE leverages Weibull advantages, namely the fact that these distributions are known to be a good representation for survival time distribution and it also allows us to consider any time horizon since we indirectly learn a continuous probability density function (through parameters learning) and thanks to the *Sparse Weibull Mixture* layer tries to select the requisite number of "Weibulls" composing the mixture to model the time distribution. We verified through simulations that DPWTE manages to model the relationship between the features and the time distribution. Regarding the experiments on real-world datasets, DPWTE has outperformed the alternative approaches. Furthermore, we assessed the censoring sensitivity of our model with both simulated-data and real-data experiments, and these demonstrate its ability to generally handle highly censored setting and consider any survival time horizon. Interesting expansions include extending our methodology to models that handle competing events, time-dependent covariates. In addition, it would be interesting to explore other data types and sources that require some advanced network structures notably convolutions neural networks or generative adversarial models.

References

[1] C. D.R., Regression models and life tables (with discussion), Journal of the Royal Statistical Society. Series B. 34 (1972) 187–220.

[2] D. Faraggi, R. Simon, A neural network model for survival data, Statistics in medicine 14 (1) (1995) 73–82.

[3] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, Y. Kluger, Deep survival: A deep cox proportional hazards network, stat 1050 (2016) 2.

[4] F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, R. A. Rosati, Evaluating the yield of medical tests, Jama 247 (18) (1982) 2543–2546.

[5] H. Kvamme, Ø. Borgan, I. Scheel, Time-to-event prediction with neural networks and cox regression, Journal of Machine Learning Research 20 (129) (2019) 1–30.

[6] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, M. S. Lauer, et al., Random survival forests, The annals of applied statistics 2 (3) (2008) 841–860.

[7] S. Yousefi, F. Amrollahi, M. Amgad, C. Dong, J. E. Lewis, C. Song, D. A. Gutman, S. H. Halani, J. E. V. Vega, D. J. Brat, et al., Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models, Scientific reports 7 (1) (2017) 1–11.

[8] X. Zhu, J. Yao, J. Huang, Deep convolutional neural network for survival analysis with pathological images, in: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2016, pp. 544–547.

[9] X. Zhu, J. Yao, F. Zhu, J. Huang, Wsisa: Making survival prediction from whole slide histopathological images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7234–7242.

[10] C. Lee, W. R. Zame, J. Yoon, M. van der Schaar, Deephit: A deep learning approach to survival analysis with competing risks, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. x–y.

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

[12] E. Martinsson, Weibull time to event recurrent neural network, Doctoral dissertation, Chalmers University of Technology & University of Gothenburg (2016).

[13] A. Bennis, S. Mouysset, M. Serrurier, Estimation of conditional mixture weibull distribution with right censored data using neural network for time-to-event analysis, Pacific-Asia Conference on Knowledge Discovery and Data Mining (2020).

[14] C. Louizos, M. Welling, D. P. Kingma, Learning sparse neural networks through l_0 regularization, arXiv preprint arXiv:1712.01312 (2017).

[15] Z. Xu, H. Zhang, Y. Wang, X. Chang, Y. Liang, L 1/2 regularization, Science China Information Sciences 53 (6) (2010) 1159–1169.

[16] J. Fan, H. Peng, et al., Nonconcave penalized likelihood with a diverging number of parameters, The Annals of Statistics 32 (3) (2004) 928–961.

[17] S. R. P. National Cancer Institute, DCCPS, Surveillance, epidemiology, and end results (seer) program (www.seer.cancer.gov) seer*stat database: Incidence - seer 18 regs research data + hurricane katrina impacted louisiana cases, nov 2018 sub (1975-2016 varying), linked To County Attributes - Total U.S., 1969-2017 Counties, released April 2019, based on the November 2018 submission. (2019).