Deep Learning-Based Discrete Calibrated Survival Prediction

Supplemental Material

Patrick Fuhlert, Anne Ernst, Esther Dietrich, Fabian Westhaeusser, Karin Kloiber, Stefan Bonn
Institute of Medical Systems Biology, Center for Biomedical AI (bAIome), Center for Molecular Neurobiology (ZMNH)
University Medical Center Hamburg-Eppendorf, Hamburg, Germany

S-1. Estimating the number of comparisons for C-index-td and newly proposed loss

Suppose a dataset D with n individuals and a censoring rate of $c \in [0, 1)$. When drawing two individuals i and j from D at random, under the assumption that censoring is uniformly distributed throughout the observed event time, the chance of picking a pair (i, j) that is comparable regarding the C-index-td (7) or $\tilde{\mathcal{L}}_{kernel}$ (5) (EE or EC pair) is

$$\begin{aligned} P(\text{comp}_{\text{new}}) =& P(d_i = 1, d_j = 1, z_i < z_j) + \\ & P(d_i = 1, d_j = 0, z_i < z_j) \end{aligned} \tag{S-1}$$

and can be estimated by using the censoring rate c of randomly drawing a censored individual from the dataset. Under the assumption that, when picking a random pair (i, j), the corresponding event times are equally distributed, the probability that the first event time z_i is smaller than z_j is assumed as $P(z_i < z_j) = 0.5$. Firstly, the relative frequency of drawing only EE pairs which corresponds to the number of comparisons in (3) can be estimated as

$$P(\text{comp}_{\text{old}}) = P(d_i = 1, d_j = 1, z_i < z_j) = (1 - c)^2/2. \tag{S-2}$$

Furthermore, we add the comparisons where the former event time is uncensored and the latter event time is censored as

$$P(d_i = 1, d_j = 0, z_i < z_j) = c(1 - c)/2. \tag{S-3}$$

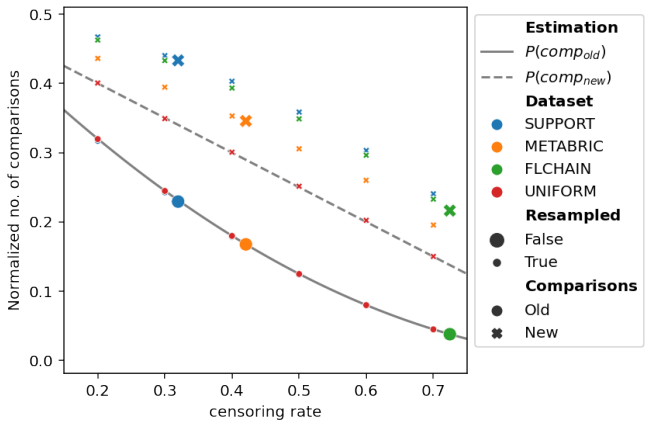
Combining (S-2) and (S-3) leads to the final estimation for the new number of comparisons

$$P(\text{comp}_{\text{new}}) = (1 - c)^2/2 + c(1 - c)/2 = (1 - c)/2. \tag{S-4}$$

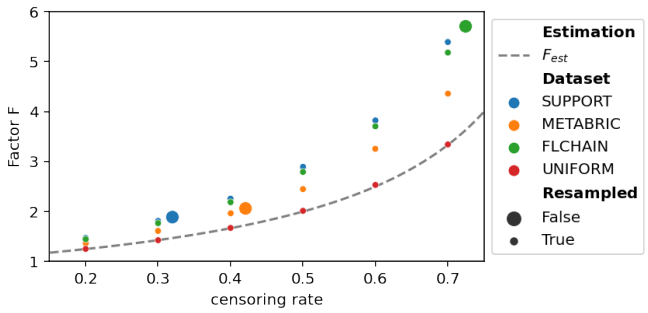
Since the old $\mathcal{L}_{\text{kernel}}$ (3) only compared all uncensored pairs for i and j that led to $P(\text{comp}_{old}) = (1-c)^2$, the factor F of more comparisons can be estimated from (S-2) and (S-4) as

$$F_{\text{est}} = P(\text{comp}_{\text{new}})/P(\text{comp}_{\text{old}}) = 1/(1 - c). \tag{S-5}$$

This estimation can now be compared to the observed number of comparisons from Tab. I in Fig. S-1. It is shown that our estimation closely matches the number of old comparisons for all datasets as well as the number of new comparisons for the synthetically created new "uniform" dataset. It is shown that, for higher censoring rates, the factor F suggests that our approach allows for significantly more comparisons on datasets with high censoring rates (e.g. 70 % censoring leads to approximately 3 to 6 times more comparisons). Also notice that the observed factor F is even higher than predicted for the real-world datasets which indicates that the assumption of a uniform censoring distribution is not met.



(a) Number of normalized comparisons n_{comp}/n^2 and estimations (S-2, S-4) over the censoring rate. Old comparisons are depicted with "o" while new comparisons are shown with an "x".



(b) Factor F (S-5) of more comparisons over the censoring rate.

Fig. S-1. Observed (markers) and estimated (lines) number of comparisons over the censoring rate for the analyzed datasets. We also include a synthetic dataset ('uniform') in red with a uniform censoring distribution. To generate additional datapoints, the datasets are resampled to match predefined censoring rates that are depicted with smaller markers.

S-2. Hyperparameter Tuning

This work utilizes a bayesian hyperparameter search with `scikit-optimize`. The data was split into 80% train and 20% test data stratified by the event indicator. Afterwards, 5-fold cross-validation on the training set was performed with 100 iterations to find the best hyperparameters. The CDAUC was chosen as the optimization criterion. The best model was then retrained on the complete training dataset with the previously found best hyperparameters. We used a fixed batch size of 50 and trained all models for 100 epochs. To avoid overfitting during training, we added early stopping with a patience of 10 epochs and included a 20 % dropout rate.

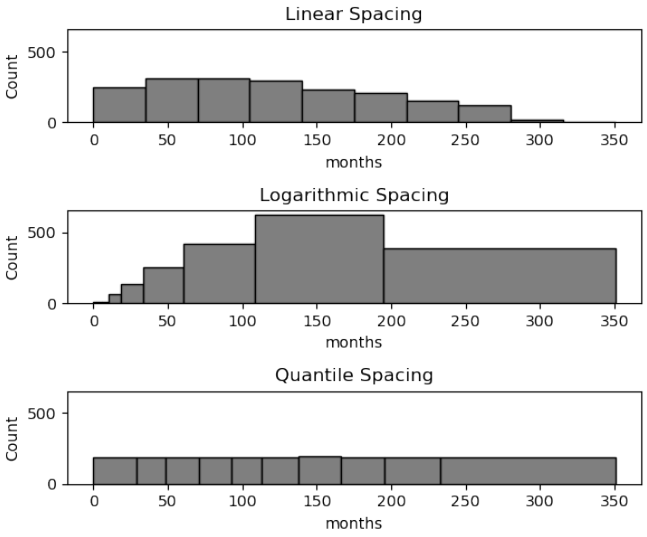
The tunable hyperparameters of the DeepSurv and CoxTime models were the number of layers and nodes per layer of the networks. CoxPH did not have any tunable hyperparameters. In DRSA, the loss weighting parameter α was also included as a hyperparameter.

All models that are based on the DRSA architecture, namely DRSA, Kamran, and DCS, share the following tunable hyperparameters. The *encoder* part represents the dense network structure before the LSTM, the *decoder* the LSTM structure itself and *aggregation* the dense network that converts each of the decoder outputs into a scalar value. Each subnetwork's number of layers and number of nodes were hyperparameter tuned. Moreover, we introduce the observation window T_{max} that corresponds to the maximum survival time in months for each training dataset. We then use the data-driven approach of defining the total number of output nodes L as a hyperparameter based on T_{max}, such that $L \in \{.25, .5, 1, 2\} \cdot T_{\text{max}}$. Note that this approach only sets the number of output nodes L of the network and is independent of the temporal spacing (linear, logarithmic or quantile) of those nodes. Furthermore, instead of fixing the loss weighting parameter λ to 1 as in the original Kamran model, it is also included together with σ (5) in the hyperparameter search space for Kamran and all DCS models.

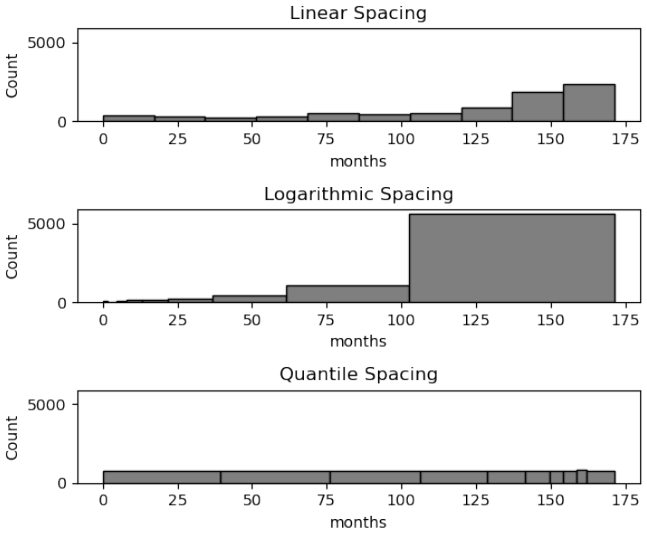
Table S-1 shows the best hyperparameters per model for each dataset. For all discrete-time models, the optimal network architectures tend to be rather shallow. In some cases, some parts of the architecture, namely encoder, decoder, or aggregation are dismissed. Notice that DeepSurv and CoxTime almost always have best performance with shallow neural networks – usually with one hidden layer, even though the search space included up to five. For the aggregation part of the network, one fully connected layer is necessary to map the decoder's output to a single hazard rate h_l. In our approach, we allow additional fully connected aggregation layers for a deeper model architecture. We also analyzed the influence of the total number of output nodes in the hyperparameter tuning. Here we can see incoherent results that might depend on the dataset as well as the temporal spacing of output nodes. In some cases, increasing the number of output nodes yields the best results (e.g. DCS-linear, DCS-log on SUPPORT and METABRIC), in others (e.g. DCS-quant on SUPPORT and METABRIC) less total output nodes L were selected.

S-3. Event histograms of other datasets

Histograms of the number of events per temporal prediction for the METABRIC and FLCHAIN datasets are shown in Fig. S-3.



(a) METABRIC



(b) FLCHAIN

Fig. S-2. Histograms of the number of events per temporal prediction on METABRIC and FLCHAIN. Ten buckets in time are made with linear, logarithmic and quantile temporal spacing.

TABLE S-1
Best hyperparameters for all analyzed datasets and the presented models.

SUPPORT	DeepSurv	CoxTime	DRSA	Kamran	DCS-linear	DCS-log	DCS-quant
encoder_num_layers	1	1	0	0	1	2	2
encoder_nodes_per_layer	64	32	-	-	32	128	64
decoder_num_layers			1	1	1	0	1
decoder_nodes_per_layer			128	32	32	-	64
decoder_bidirectional					0	-	1
decoder_use_lstm_skip					0	-	0
aggregation_num_layers				1	1	1	2
additional_aggregation_nodes_per_layer				-	-	-	32
output_grid_num_nodes			280	67	140	140	35
α			0.36				
λ				2.00	0.85	0.25	2.00
σ				1.08	1.55	2.00	1.00

METABRIC	DeepSurv	CoxTime	DRSA	Kamran	DCS-linear	DCS-log	DCS-quant
encoder_num_layers	1	1	0	0	2	1	0
encoder_nodes_per_layer	8	8	-	-	128	64	-
decoder_num_layers			1	1	2	1	2
decoder_nodes_per_layer			128	64	64	64	32
decoder_bidirectional					0	1	0
decoder_use_lstm_skip					1	1	0
aggregation_num_layers				1	2	1	1
additional_aggregation_nodes_per_layer				-	32	-	-
output_grid_num_nodes			350	350	700	350	60
α			0.03				
λ				1.94	1.19	1.08	0.25
σ				1.63	0.25	1.45	2.00

FLCHAIN	DeepSurv	CoxTime	DRSA	Kamran	DCS-linear	DCS-log	DCS-quant
encoder_num_layers	5	1	1	0	1	0	1
encoder_nodes_per_layer	64	8	64	-	64	-	64
decoder_num_layers			1	1	1	1	0
decoder_nodes_per_layer			16	32	64	128	-
decoder_bidirectional					1	1	-
decoder_use_lstm_skip					1	0	-
aggregation_num_layers				1	1	1	1
additional_aggregation_nodes_per_layer				-	-	-	-
output_grid_num_nodes			42	171	125	42	85
α			0.39				
λ				1.34	2.00	0.67	1.13
σ				0.34	1.00	0.46	0.73