A Monte Carlo Framework for Calibrated Uncertainty Estimation in Sequence Prediction

Anonymous authors
Paper under double-blind review

Abstract

Probabilistic prediction of sequences from images and other high-dimensional data remains a key challenge, particularly in safety-critical domains. In these settings, it is often desirable to quantify the uncertainty associated with the prediction (instead of just determining the most likely sequence, as in language modeling). In this paper, we consider a Monte Carlo framework to estimate probabilities and confidence intervals associated with sequences. The framework uses a Monte Carlo simulator, implemented as an autoregressively trained neural network, to sample sequences conditioned on an image input. We then use these samples to estimate probabilities and confidence intervals. Experiments on synthetic and real data show that the framework produces accurate discriminative predictions, but can suffer from miscalibration. To address this shortcoming, we propose a time-dependent regularization method, which produces calibrated predictions.

1 Introduction

We consider the problem of predicting a sequence of multi-class labels from high-dimensional input data, such as images. Potential applications include patient prognostics from medical imaging data (Pham et al., 2017), weather forecasting (Andrychowicz et al., 2023), and modeling of human behavior (Thakkar et al., 2024). Our focus is on *probabilistic* prediction, which requires uncertainty quantification in the form of probabilities or confidence intervals.

Figure 1 shows two examples of sequence prediction problems: modeling player behavior in a video game and health monitoring. The data consist of "starting state" images and subsequent sequences of player actions (e.g., *move left*, *move right*, *shoot*, etc.) and patients' health statuses (*healthy*, *ill* or *dead*), respectively. The goal is to predict the sequence of actions/health statuses from the starting state image.

There has been a large volume of work on estimating the *most likely* sequence from high-dimensional data, for example, in language modeling (You et al., 2016; Herdade et al., 2019; Li et al., 2022; Brown et al., 2020). However, predicting a most likely sequence is *not sufficient* when the evolution of the sequence is uncertain. In this case, multiple states are possible at a future time given a fixed input, and therefore a single deterministic estimate is not acceptable. Instead, we need to predict the *probability* that each state occurs at a given point in the future. Similarly, in prediction of time-to-event values (e.g., the time until the game ends, or the survival time of a subject), predicting the most likely value is also insufficient, as there is a range of possible times given a specific input. Instead, we wish to produce confidence intervals that are likely to contain the time of interest with high probability.

Predicting the probability of the different possible states of a sequence at a fixed future time and survival modeling are standard classification and probability-estimation problems (Liu et al., 2022). However, existing approaches require separate models for each time and type of event. In this paper, we propose a framework that uses a single model to generate probabilities and confidence intervals associated with any possible future time and any type of event.

The proposed **f**ramework for M**o**nte **C**arlo **u**ncertainty quantification of **s**equences (**foCus**) combines autoregressive models—which are the state of the art for many sequence-based tasks (Bahdanau et al., 2014;

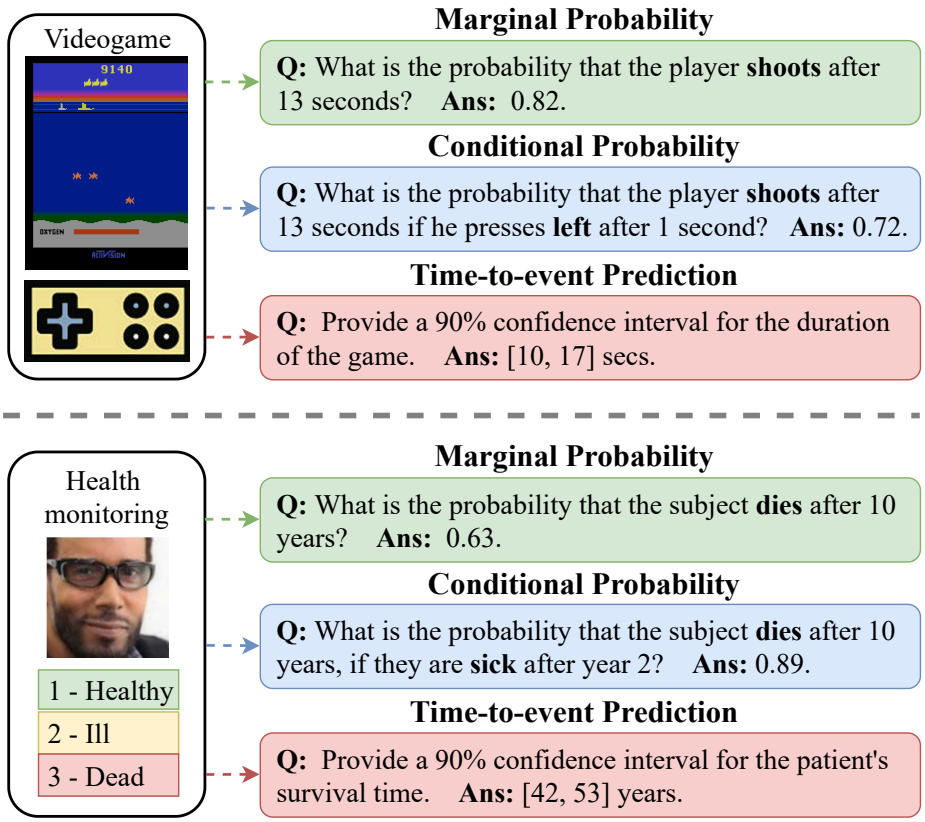


Figure 1: **Sequence prediction with uncertainty estimation.** The proposed framework enables estimation of marginal probabilities, conditional probabilities, and time-to-event confidence intervals associated with a sequence given an input image. We consider sequential decision-making tasks where the input image is a screenshot from an Atari video game and the sequence to predict is the sequence of actions taken by a user (top), and we consider a synthetic-data forecasting problem where the input image is the face of a person and the sequence to predict is the evolution of their health status (bottom).

Gehring et al., 2017; Vaswani et al., 2017; Brown et al., 2020), with Monte Carlo estimation to produce useful uncertainty estimates for sequence prediction tasks. However, when studying this framework, we find that autoregressive simulators trained via maximum likelihood estimation are severely miscalibrated, meaning that the associated probabilities or confidence intervals do not provide accurate uncertainty quantification for the tasks at hand. To address this shortcoming, we develop a time-dependent regularizer for autoregressive simulators training and show that it enables foCus to generate better calibrated probability estimates.

To summarize, our main contributions are as follows:

1. We propose a Monte Carlo framework for probabilistic prediction of sequences from high-dimensional input data using autoregressive models.
2. We perform an empirical evaluation of the proposed framework on a hand-tailored synthetic benchmarking task for sequence prediction and on non-synthetic sequential decision-making tasks and find that neural network-based autoregressive simulators are prone to time-dependent miscalibration.
3. We develop a time-dependent regularization method and show that it allows learning simulators that produce calibrated uncertainty estimates under the proposed framework.

2 Related Work

Sequence generation. Discrete sequence generation plays a fundamental role in many natural language processing applications, such as language modeling (Brown et al., 2020; Touvron et al., 2023), image captioning (Ghandi et al., 2023), language translation (Bahdanau et al., 2014; Gehring et al., 2017; Vaswani et al., 2017), and text summarization (Dong, 2018). In these examples, sequence generation is typically performed in an autoregressive manner, where each token is generated based on the tokens previously generated. Alternatively, sequences can also be generated non-autoregressively (Sun & Yang, 2020; Gu et al., 2017; Shu et al., 2020), where tokens are produced simultaneously or with fewer dependencies on earlier tokens. In both paradigms, the primary objective is to generate the most likely sequence. However, these approaches typically do not focus on quantifying the uncertainty associated with generated sequences, which is the goal of this paper.

Imitation and reinforcement learning. Our focus is on *predicting* sequences from high-dimensional data, which is fundamentally different from imitation learning (Hussein et al., 2017), which seeks to replicate human behavior, and reinforcement learning (Sutton, 2018), which seeks to determine an optimal policy by allowing the agent to interact with the environment, guided by a reward function (Schulman et al., 2017; Lillicrap et al., 2019).

Calibration. Miscalibration is a well-known challenge in classification models based on deep learning (Guo et al., 2017; Wang, 2024; Wang et al., 2021), particularly when the goal is to provide accurate uncertainty quantification (Liu et al., 2022). While significant progress has been made in calibrating classification models, calibration in sequence prediction tasks remains underexplored and even lacks a clear definition. While Marx et al. (2024) explores calibration in sequences, their focus is on step-wise calibration, where each sequence step has an input-output pair, which makes the problem step-wise calibration within sequences, whereas our work focuses on calibration for the entire sequence. Kuleshov & Liang (2015) offers a framework for measuring calibration in structured high-dimensional random vectors via event pooling, which inspires our approach to uncertainty estimation in sequence prediction.

Various methods have been developed to enhance calibration in classification, including post-processing the logits (Gupta et al., 2021; Kull et al., 2017; 2019), ensembling methods (Zhang et al., 2020a; Lakshminarayanan et al., 2017; Maddox et al., 2019), soft labeling (Mukhoti et al., 2020; Szegedy et al., 2016; Zhang et al., 2017a; Thulasidasan et al., 2019; Liu et al., 2022) and training with regularization (Pereyra et al., 2017; Kumar et al., 2018; Rudner et al., 2023). In this work, we propose a time-dependent regularization method to improve calibration specifically in sequence prediction tasks.

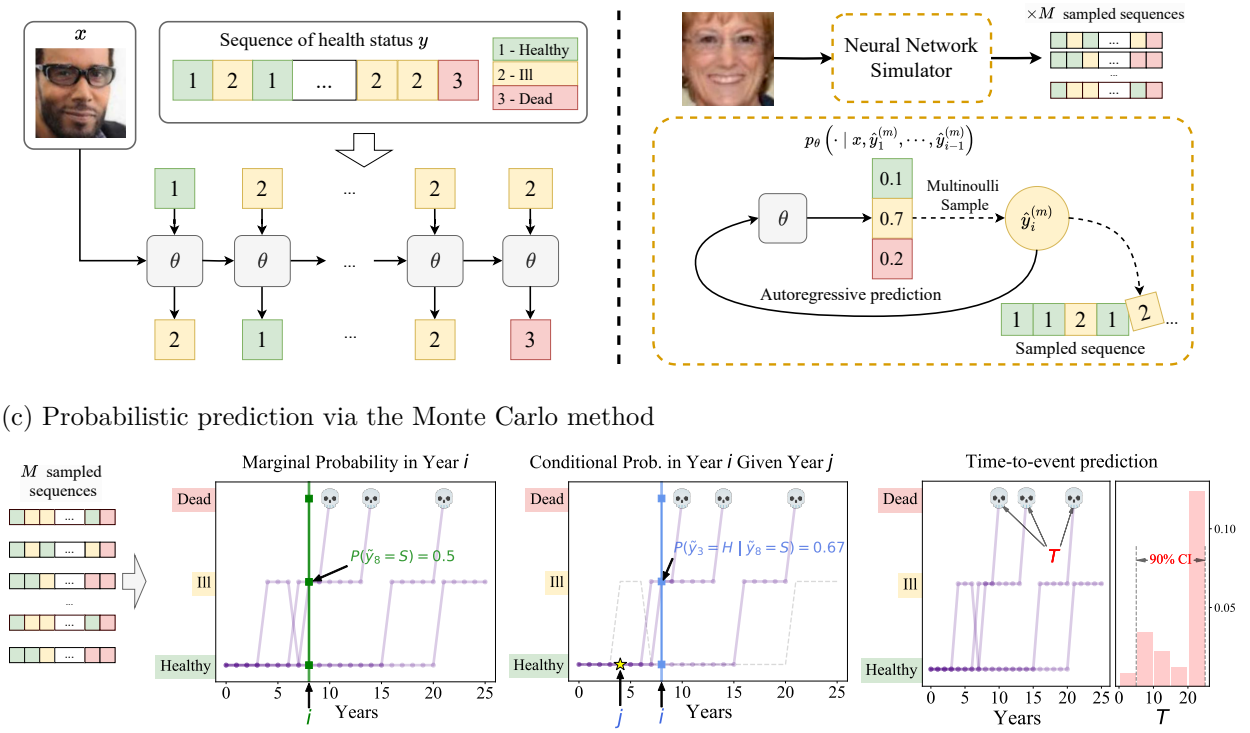
The Monte Carlo method in deep learning. The Monte Carlo method is used for generating ensembles in Bayesian deep learning from samples of the model parameters (Blundell et al., 2015; Gal & Ghahramani, 2016). In contrast, in our proposed framework we sample sequences using a neural-network simulator with fixed parameters, and apply the Monte Carlo method to the sampled sequences. Monte Carlo methods are also used for uncertainty estimation on large language model outputs (Malinin & Gales, 2021; Jiang et al., 2021; Kuhn et al., 2023; Xiong et al., 2024), where multiple sequences are generated to assess the confidence of factual outputs and mitigate hallucinations. Unlike these works, our goal here is not to determine the most likely sequence, but rather to provide a probabilistically-accurate characterization of possible future sequences.

3 A Monte Carlo Framework

We consider the problem of predicting a sequence of multi-class labels from high-dimensional data. More formally, our goal is to estimate the conditional distribution of a sequence Y consisting of ℓ discrete random variables each with c possible states given an observed input x, interpreted as a sample of a random vector X. In our health-monitoring example, the $c := 3$ states are *healthy*, *ill* and *dead*, and X represents an input image.

(a) Training of the autoregressive simulator

(b) Sampling from the simulator given an input image



(c) Probabilistic prediction via the Monte Carlo method

Figure 2: **Monte Carlo framework for uncertainty estimation in sequence prediction.** (a) A neural network simulator is trained to autoregressively predict the conditional distribution of each entry in a sequence given an input image and the preceding states. (b) The simulator is used to generate multiple sample sequences, by iteratively sampling from the estimated conditional distribution. (c) The Monte Carlo method is applied to estimate marginal probabilities, conditional probabilities, and time-to-event confidence intervals from the samples.

Even for short sequence lengths, directly estimating the joint conditional probability mass function of Y given $X = x$ is intractable due to the combinatorial explosion of possible sequences (e.g., for $c := 3$ and $\ell := 100$ there are $3^{100} > 10^{47}$ possible sequences!), which is an instance of the notorious curse of dimensionality. Instead, we propose to estimate the following probabilities and confidence intervals characterizing the sequence, which are illustrated in Figure 1:

1. The *marginal* probability $\mathrm{P}(Y_i = a \,|\, X = x)$ that the ith entry Y_i of the sequence is equal to $a \in \{1, ..., c\}$. We refer to this as a marginal probability, but—strictly speaking—it is a conditional probability, as it is conditioned on $X = x$. In health monitoring, this is the probability that a subject is healthy, ill, or dead at time i.

2. The *conditional* probability $\mathrm{P}(Y_i = a \,|\, Y_j = b, X = x)$ that the ith entry Y_i of the sequence is equal to $a \in \{1, ..., c\}$ given that the jth entry is equal to $b \in \{1, ..., c\}$. In health monitoring, this could be the conditional probability that a subject is dead at time i given that they are ill at time j.

3. The α *confidence interval* I_α for the time $\widetilde{T}$ until a certain event associated with the sequence occurs (e.g. $\widetilde{T} := \min\{i \,:\, Y_i = a\}$ for some $a \in \{1, ..., c\}$), which should satisfy $\mathrm{P}(\widetilde{T} \in I_\alpha) = \alpha$, where α is typically set to 0.9 or 0.95. For example, in health monitoring, $\widetilde{T}$ can represent the time of death or recovery.

3.1 Monte Carlo Estimation

As noted in the previous section, a key challenge in estimating arbitrary probabilities and confidence intervals associated with a sequence of random variables Y is that it is intractable to *explicitly* estimate the sequence joint distribution given the input X (unless we make highly simplifying modeling assumptions, such that the sequence forms a Markov chain). Our proposed **f**ramework for M**o**nte **C**arlo **u**ncertainty quantification of **s**equences (**foCus**) addresses this challenge by instead *implicitly sampling* from the conditional distribution using a neural network simulator, described in Section 3.2.

Given an input x, we apply the simulator to generate M sequences $\{(\hat{y}_1^{(m)}, ..., \hat{y}_l^{(m)})\}_{m=1}^{M}$. As shown in panel (c) of Figure 2, the sequences are then used to estimate any desired probability or confidence interval. The marginal probability of state a at time i is estimated by the fraction of sequences in state a at time i:

$$\mathrm{P}(Y_i = a \,|\, X = x) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}_{\{\hat{y}_i^{(m)} = a\}}. \tag{1}$$

The conditional probability of state a at time i given state b at time j is estimated by the fraction of sequences in state a at time i out of sequences in state b at time j:

$$\mathrm{P}(Y_i = a \,|\, Y_j = b, X = x) = \frac{\sum_{m=1}^{M} \mathbb{1}_{\{\hat{y}_i^{(m)} = a, \hat{y}_j^{(m)} = b\}}}{\sum_{m=1}^{M} \mathbb{1}_{\{\hat{y}_j^{(m)} = b\}}}. \tag{2}$$

To estimate the α confidence interval I_α of the time-to-event $\widetilde{T}$, we first compute the value of $\widetilde{T}$ associated with each simulated sequence. These yields M times $\{T^{(1)}, ..., T^{(M)}\}$, which can be sorted to calculate the $(1-\alpha)/2$ and $(1+\alpha)/2$ percentiles $q_{(1-\alpha)/2}$ and $q_{(1+\alpha)/2}$. The confidence interval I_α is set to $[q_{(1-\alpha)/2}, q_{(1+\alpha)/2}]$.

3.2 Autoregressive Simulation

In order to produce the simulated sequences required by our Monte Carlo framework, we employ a neural-network simulator. The simulator consists of a convolutional neural network (CNN) that encodes the input image x, and a recurrent neural network (RNN) that iteratively estimates the conditional distribution of the ith entry Y_i of the sequence given $X = x$ and the values of all previous entries, i.e. $\mathrm{P}\left(Y_i = y_i \mid X = x, Y_1 = y_1, ..., Y_{i-1} = y_{i-1}\right)$.

Figure 2(b) illustrates how the simulator is used to obtain a sample sequence $(\hat{y}_1, \ldots, \hat{y}_\ell)$. The input image x is fed into the CNN, producing a hidden vector h_0 that is fed into the RNN to then generate the simulated sequence iteratively. At each iteration $i \in \{1, ..., \ell\}$, the input of the RNN is the value $\hat{y}_{i-1}$ of the previous entry (except for $i = 1$) and the hidden vector h_{i-1}. The outputs are an estimate of the conditional distribution of Y_i given the previous entries and $X = x$, and an updated hidden vector h_i. The ith entry $\hat{y}_i$ of the sample sequence is sampled from this conditional distribution.

The simulator is trained using a dataset of image-sequence pairs, as illustrated in Figure 2(a). During training, the model uses the ground truth value y_{i-1} of the previous entry along with the hidden vector h_{i-1} to predict the subsequent ith entry Y_i. As explained in Sections 5 and 6 the design of the training loss plays a crucial role in avoiding miscalibration in the downstream probabilities and confidence intervals computed using the simulator.

4 Evaluating Uncertainty

4.1 Evaluation metrics

We assess marginal and conditional probability estimates with a set of complementary metrics. Macro Area Under the ROC Curve (AUC) quantifies discriminative ability. Expected Calibration Error (ECE) quantifies

Table 1: **Marginal probability estimation.** The table reports sequence-level metrics evaluating the performance of the proposed foCus framework for estimation of marginal probabilities (see Section 3). We compare versions of foCus without regularization (see Section 5) and with constant and time-dependent regularization (see Section 6). Results are presented as mean ± standard error from three independent model realizations. Time-dependent regularization improves calibration substantially (lower ECE), while maintaining a comparable AUC, which results in superior probability estimates (lower cross entropy and Brier score). Similar results are obtained for conditional probability estimation, as reported in Table 4.

Scenario	Regularization	ECE (↓)	AUC (↑)	CE (↓)	BS (↓)
Seaquest	✗	0.0435 ± 0.0004	0.8671 ± 0.0035	1.0577 ± 0.0241	0.1247 ± 0.0012
	time-dependent	**0.0277 ± 0.0023**	**0.8678 ± 0.0028**	**0.6705 ± 0.0285**	**0.1144 ± 0.0007**
	constant	0.0365 ± 0.0002	0.8625 ± 0.0020	0.8173 ± 0.0068	0.1177 ± 0.0008
River Raid	✗	0.0583 ± 0.0016	**0.6453 ± 0.0009**	1.2034 ± 0.0281	0.1750 ± 0.0016
	time-dependent	**0.0388 ± 0.0013**	0.6346 ± 0.0035	**0.8585 ± 0.0132**	**0.1671 ± 0.0012**
	constant	0.0474 ± 0.0004	0.6280 ± 0.0020	1.0274 ± 0.0158	0.1686 ± 0.0005
Bank Heist	✗	0.0559 ± 0.0032	**0.6938 ± 0.0028**	1.1874 ± 0.0540	0.2340 ± 0.0020
	time-dependent	**0.0148 ± 0.0014**	0.6782 ± 0.0016	**0.7647 ± 0.0130**	**0.2166 ± 0.0005**
	constant	0.0399 ± 0.0016	0.6928 ± 0.0046	0.8894 ± 0.0112	0.2211 ± 0.0007
H.E.R.O.	✗	0.0947 ± 0.0014	0.6785 ± 0.0061	1.1310 ± 0.0225	0.1261 ± 0.0009
	time-dependent	0.0481 ± 0.0034	**0.7159 ± 0.0105**	**0.6940 ± 0.0246**	**0.1170 ± 0.0008**
	constant	**0.0352 ± 0.0001**	0.7041 ± 0.0102	0.7218 ± 0.0391	0.1212 ± 0.0011
Road Runner	✗	0.0779 ± 0.0035	**0.6913 ± 0.0100**	1.1586 ± 0.0291	0.1575 ± 0.0034
	time-dependent	**0.0204 ± 0.0012**	0.6823 ± 0.0084	**0.5255 ± 0.0077**	**0.1382 ± 0.0003**
	constant	0.0303 ± 0.0027	0.6898 ± 0.0140	0.6275 ± 0.0250	0.1394 ± 0.0010
FaceMed	✗	0.1503 ± 0.0048	0.7534 ± 0.0079	1.6932 ± 0.0188	0.3464 ± 0.0012
	time-dependent	**0.0757 ± 0.0068**	**0.7614 ± 0.0024**	**0.9085 ± 0.0303**	**0.3328 ± 0.0008**
	constant	0.0974 ± 0.0045	0.7613 ± 0.0028	1.0071 ± 0.0499	0.3356 ± 0.0030

calibration. Brier Score (BS) and Cross Entropy (CE) provide a more holistic evaluation. These metrics are computed for each entry and then aggregated via averaging to obtain sequence-level metrics.

To assess confidence intervals, we evaluate discriminability via the Mean Absolute Error (MAE) (normalized by the average sequence length). Calibration is evaluated by computing the coverage probability of the intervals, which should be close to the target confidence level α. We also measure the average confidence interval width relative to the average sequence length. Additional details are provided in Appendix A.

4.2 Datasets

FaceMed is a synthetic dataset designed to predict individual health status trajectories based UTKFace (Zhang et al., 2017b), which contains of real face images of subjects with different ages. For each image, we simulate a sequence of health states using a Markov chain model that depends on the age of the subject (see Appendix B). The health states are *healthy*, *ill*, and *dead*. The goal is to predict the future marginal and conditional probability distribution of health status in the future, as well as confidence intervals for subject survival.

Atari games is a non-synthetic benchmark consisting of five real human gameplay datasets from Atari-HEAD (Zhang et al., 2020b), consisting of screenshots associated with subsequent sequences of player actions. There are 19 actions, including *move left*, *fire*, and an *end* action, which indicates the end of a game sequence. The games are:

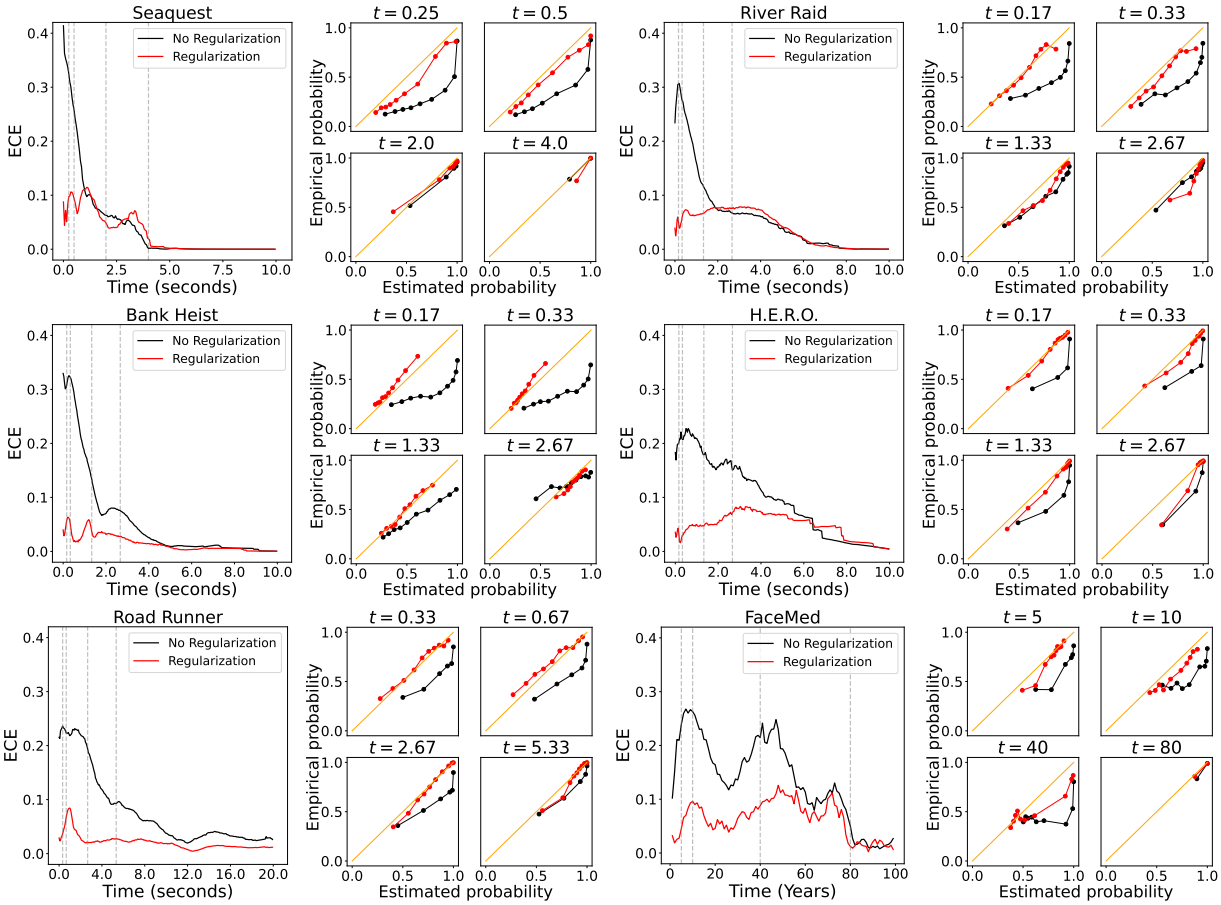


Figure 3: **Entry-wise calibration error and reliability diagrams for marginal probability estimation.** The large graphs plot the entry-level ECE of the proposed foCus framework for estimation of marginal probabilities (see Section 3) without regularization (black line, see Section 5) and time-dependent regularization (red line, see Section 6). Unregularized foCus produces miscalibrated estimates, particularly in the earlier entries, which are dramatically improved by time-dependent regularization for all datasets. The small graph show reliability diagrams for some of the steps, which confirm the improvement in calibration. Additional reliability diagrams and results for constant regularization are shown in Appendix D.1.

- Seaquest: Control a submarine to rescue divers while shooting sharks and enemy submarines;
- River Raid: Navigate a fighter jet to destroy enemy targets while managing fuel;
- Bank Heist: Drive through a maze-like city to rob banks while avoiding police;
- H.E.R.O.: Traverse a mineshaft to rescue trapped miners while avoiding enemies and hazards;
- Road Runner: Guide a bird to collect seeds while evading a chasing coyote and obstacles.

The goal is to estimate the marginal and conditional probabilities of player actions, and confidence intervals of the time until the player scores. All datasets are split into training, validation and test sets following a 7:2:1 ratio. Additional details about datasets are provided in Appendix B.

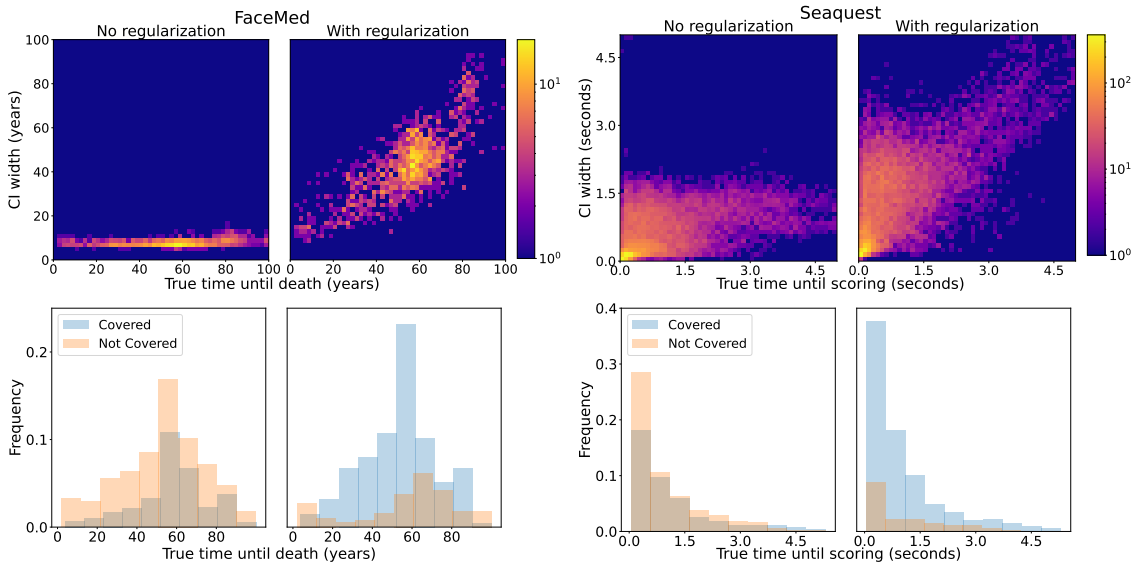


Figure 4: **Confidence intervals for time-to-event prediction and coverage probability.** The upper panel shows heatmaps of the length of 0.9 confidence intervals for time-to-event prediction using the proposed foCus framework without and with time-dependent regularization. The histograms below show the frequency of intervals containing the true times, as a function of the true time. Unregularized foCus produces short intervals with poor coverage, whereas regularization yields intervals that tend to be larger when the ground-truth times are larger, and are much better calibrated. The plots correspond to the FaceMed (left) and Seaquest (right) datasets. Appendix D.3 shows analogous plots for the remaining datasets.

5 Miscalibration

In this section, we report the results of applying foCus when we train the simulator described in Section 3.2 using a standard unregularized cross-entropy loss

$$\mathrm{CE}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[-\sum_{i=1}^{\ell} \log p_\theta \left(y_i \mid x, y_1, ..., y_{i-1} \right) \right], \tag{3}$$

where x is an input image, y is the corresponding sequence and $\mathcal{D}$ is the training set of image-sequence pairs. Here θ represents the parameters in the neural-network simulator and $p_\theta \left(y_i \mid x, y_1, ..., y_{i-1} \right)$ denotes the corresponding estimate of the conditional probability $\mathrm{P}\left(Y_i = y_i \mid X = x, Y_1 = y_1, ..., Y_{i-1} = y_{i-1} \right)$. Further details about the training procedure are provided in Appendix C.2.

Tables 1 and 4 show that this version of foCus has strong discriminative performance across all our datasets when estimating marginal and conditional probabilities, respectively, indicated by the high sequence-level AUC values. However, the sequence-level ECE values are also high, suggesting that the probability estimates are not well calibrated. This is corroborated by Figure 3, which shows entry-level ECE values for all datasets. Miscalibration is time-dependent and particularly severe at the beginning of the sequence. The reliability diagrams in Figure 3 show that the model suffer from *overconfidence* (the estimated probabilities are more extreme than the empirical probabilities), which is typical of deep learning models, as they tend to overfit the training labels (Liu et al., 2022).

We also observe miscalibration in the confidence intervals estimated by this version of foCus. Table 5 in Appendix D.3 reports the coverage probabilities computed on test data for $\alpha = 0.9$, which are below 0.5 in all cases except one! Moreover, Figure 4 reveals that the widths of the confidence intervals remain invariant over time, which is undesirable as uncertainty should increase with longer time horizons.

6 Time-dependent Logit Regularization

To address simulator-induced miscalibration in foCus, we incorporate a regularization term that penalizes the ℓ_2-norm of the logits within the simulator. This regularizer is motivated by recent work on function-space regularization (Rudner et al., 2023; 2024a;b; Klarner et al., 2024), which interprets regularized objectives as performing maximum a posteriori (MAP) estimation under data-driven-priors over neural network parameters. Intuitively, the ℓ_2-norm penalty prevents the logits from becoming very large during training. This mitigates overconfidence, since large logits result in more extreme probability estimates. We can think of the ℓ_2-norm penalty as promoting neural network parameters that induce predictive functions with higher predictive entropy. As in Section 5, let $p_\theta(y_i|x, y_1, ..., y_{i-1})$ denote the estimate of the conditional probability $\mathrm{P}\left(Y_i = y_i \mid X = x, Y_1 = y_1, ..., Y_{i-1} = y_{i-1}\right)$ produced by the simulator for $i \in \{1, ..., \ell\}$, which is obtained by feeding a logit vector $\mathbf{z}(x, y_1, ..., y_{i-1}) \in \mathbb{R}^c$ into a softmax function. The training loss is

$$\mathcal{L}(\theta) \doteq \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[\sum\nolimits_{i=1}^{\ell} -\log p_\theta(y_i|x, y_1, ..., y_{i-1}) + \lambda_i \|\mathbf{z}(x, y_1, ..., y_{i-1})\|_2\right], \tag{4}$$

where λ_i is a regularization coefficient that governs the regularization strength when predicting the ith entry of the sequence. The regularization λ_i is designed to be *time dependent*, motivated by our observation that the baseline version of foCus suffers from different degrees of miscalibration at different entries.

A crucial challenge is how to select the value of this hyperparameter, given the large dimensionality of the hyperparameter space. We propose a selection procedure, based on the observation that miscalibration in the initial entries is propagated by the autoregressive structure of the simulator (see Section 3.2). Consequently, optimizing the regularization parameter at the beginning of the sequence has more impact on the overall calibration performance of foCus (see Appendix E for additional analysis). The procedure is as follows:

1. For $1 \leq i \leq k_1$ (where k_1 is a hyperparameter) we use the sequence-level ECE of marginal probabilities (see Section 4.1) computed over validation set to iteratively select λ_i, setting $\lambda_j = 0$ for all $j > i$.
2. For $k_1 < i \leq k_2$ (where k_2 is a hyperparameter) we constrain all the parameters to equal the same constant, $\lambda_i = \lambda_{\text{all}}$, selected also based on the validation ECE.
3. For $i > k_2$ we set $\lambda_i = 0$.

We set $k_1 = 3$ and performed hyperparameter optimization to select k_2, which typically resulted in small values (see Table 3). Details about the optimal regularization parameters and a comprehensive overview of the hyperparameter search process are provided in Appendix C.4.

6.1 Results

Marginal and conditional probabilities. Table 1 compares the sequence-level evaluation metrics for marginal probability estimation of foCus with (1) no regularization as described in Section 5, (2) our proposed time-dependent regularization described in Section 6, and (3) constant regularization where the regularization parameter in Equation (4) is set to a single constant $\lambda_i = \lambda_{\text{const}}$ (determined based on validation ECE). All methods achieve similar AUCs in each dataset, indicating a similar discriminative ability. In contrast, the ECE is significantly lower for time-dependent regularization for all datasets, indicating better calibration performance. This results in better probability estimates, as evidenced by the lower cross entropy (CE) and Brier scores (BS). For FaceMed this is confirmed by comparing the estimated probabilities to the ground-truth marginal probabilities. The unregularized baseline method and constant regularization yield RMSEs of 0.1847 ± 0.0014 and 0.1754 ± 0.0028, respectively, while time-dependent regularization reduces the RMSE to 0.1720 ± 0.0009 (see Appendix D.1). The same holds for conditional probability predictions, as reported in Table 4: Time-dependent regularization again significantly improves calibration, and as a result the overall probability estimates. Appendix F provides a detailed description of the conditional probability estimates for one of the video games.

Figure 3 further demonstrates the improvement in calibration provided by time-dependent regularization. Interestingly, we observe a *calibration propagation* phenomenon, where regularizing a small number of early entries produces improved calibration across the whole sequence. For example, for H.E.R.O. regularization is applied to the first 6 entries (0.2 seconds), yet the ECE improvement is evident up until entry 150 (5 seconds).

Time-to-event confidence intervals. Table 5 compares the evaluation metrics for the confidence intervals produced by foCus, again with (1) no regularization, (2) time-dependent regularization, and (3) constant regularization. In this case, we observe a certain trade-off between discriminative performance, quantified by the relative MAE, and calibration, quantified by coverage probabilities. The MAE for models trained without or with constant regularization are consistently lower than those of time-dependent regularization, but the coverage probabilities of time-dependent regularization are a lot closer to 90% (between 69% and 92%, compared to at most 70% for the other two methods).

Figure 4 shows heatmaps of the confidence-interval widths for different ground-truth times (upper panel) for FaceMed and Seaquest (see Appendix H for additional plots), as well as a histogram with the fraction of intervals containing the ground-truth times. We observe that unregularized simulator training, foCus produces narrow confidence intervals with poor coverage across the board, whereas time-dependent regularization yields intervals that tend to be larger when the true times are larger, achieving much better coverage.

7 Discussion and Limitations

In this paper, we studied an important, yet underexplored topic: how to achieve reliable uncertainty quantification when predicting sequences from high-dimensional data. We proposed a Monte Carlo framework based on learned autoregressive simulators that enables flexible estimation of probabilities and confidence intervals. Our experiments on sequential decision-making tasks revealed that simulator models learned via maximum likelihood estimation can lead to severely miscalibrated uncertainty estimates. We showed that this shortcoming can be addressed by training the autoregressive simulator model using a time-dependent regularizer, which we find consistently leads to well-calibrated uncertainty estimates. The time-dependent regularizer is not only effective on the RNN simulator discussed in the main text; it also applies to other types of architectures such as transformer (Appendix G).

The proposed regularization is conceptually and mathematically simple but requires choosing a set of regularization coefficients $\{\lambda_i\}_{i=1}^{l}$ from a combinatorially large space, making an exhaustive search infeasible in practice. This is not unique to our approach: Real-world sequences often display non-stationary statistical properties that are difficult to model in a data-driven fashion. Nevertheless, we find that our simple coefficient selection protocol leads to significant improvement in calibration, although more sophisticated strategies could well result in further gains. Other potentially fruitful directions for future research are to perform uncertainty quantification of continuous-valued and spatiotemporal sequences in weather and climate applications—areas in which neural-network simulators are rapidly gaining popularity (Pathak et al., 2022; Kochkov et al., 2024; Subel & Zanna, 2024).

8 Conclusion

In this paper, (1) we proposed a Monte Carlo framework for estimating uncertainty measures for sequences conditioned on high-dimensional data using learned autoregressive simulators, (2) we performed experiments on a new synthetic benchmark dataset for sequence prediction and on real-world data that revealed that maximum likelihood estimation of neural network simulator models is prone to time-dependent miscalibration, and (3) we proposed a simple, time-dependent regularization method for learning autoregressive neural network simulators that leads to well-calibrated uncertainty estimation under the proposed Monte Carlo framework. We hope that this work will inform design choices in uncertainty estimation for sequence prediction models.

References

Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alexander Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations, 2023. URL https://arxiv.org/abs/2306.06079.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Yue Dong. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016. URL https://arxiv.org/abs/1506.02142.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pp. 1243–1252. PMLR, 2017.

Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. Deep learning approaches on image captioning: A review. *ACM Computing Surveys*, 56(3):1–39, 2023.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *International Conference on Learning Representations (ICLR)*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. *Advances in neural information processing systems*, 32, 2019.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL https://doi.org/10.1145/3054912.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57.

Leo Klarner, Tim G. J. Rudner, Garrett M. Morris, Charlotte Deane, and Yee Whye Teh. Domain-aware guidance for out-of-distribution molecular and protein design. In *Proceedings of the 41th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2024.

Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, Sam Hatfield, Peter Battaglia, Alvaro Sanchez-Gonzalez, Matthew Willson, Michael P. Brenner, and Stephan Hoyer. Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066, July 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07744-y. URL http://dx.doi.org/10.1038/s41586-024-07744-y.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. *Advances in Neural Information Processing Systems*, 28, 2015.

Meelis Kull, Telmo M. Silva Filho, and Peter Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11(2):5052 – 5080, 2017. doi: 10.1214/17-EJS1338SI. URL https://doi.org/10.1214/17-EJS1338SI.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pp. 2805–2814. PMLR, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models, 2022. URL https://arxiv.org/abs/2109.10282.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

Sheng Liu, Aakash Kaku, Weicheng Zhu, Matan Leibovich, Sreyas Mohan, Boyang Yu, Haoxiang Huang, Laure Zanna, Narges Razavian, Jonathan Niles-Weed, and Carlos Fernandez-Granda. Deep probability estimation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13746–13781. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/liu22f.html.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Charles Marx, Volodymyr Kuleshov, and Stefano Ermon. Calibrated probabilistic forecasts for arbitrary sequences, 2024. URL https://arxiv.org/abs/2409.19157.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022. URL https://arxiv.org/abs/2202.11214.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions, 2017. URL https://arxiv.org/abs/1701.06548.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of biomedical informatics*, 69:218–229, 2017.

Tim G. J. Rudner, Sanyam Kapoor, Shikai Qiu, and Andrew Gordon Wilson. Function-Space Regularization in Neural Networks: A Probabilistic Perspective. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2023.

Tim G. J. Rudner, Xiang Pan, Yucen Lily Li, Ravid Shwartz-Ziv, and Andrew Gordon Wilson. Fine-tuning with uncertainty-aware priors makes vision and language foundation models more reliable. In *ICML Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024a.

Tim G. J. Rudner, Ya Shi Zhang, Andrew Gordon Wilson, and Julia Kempe. Mind the gap: Improving robustness to subpopulation shifts with group-aware priors. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 2024b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pp. 8846–8853, 2020.

Adam Subel and Laure Zanna. Building ocean climate emulators. *arXiv preprint arXiv:2402.04342*, 2024.

Zhiqing Sun and Yiming Yang. An em approach to non-autoregressive conditional sequence generation. In *International Conference on Machine Learning*, pp. 9249–9258. PMLR, 2020.

Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Neerja Thakkar, Karttikeya Mangalam, Andrea Bajcsy, and Jitendra Malik. Adaptive human trajectory prediction via latent corridors, 2024. URL https://arxiv.org/abs/2312.06653.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art, 2024. URL https://arxiv.org/abs/2308.01222.

Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking calibration of deep neural networks: Do not be afraid of overconfidence. *Advances in Neural Information Processing Systems*, 34:11809–11820, 2021.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=gjeQKFxFpZ`.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4651–4659, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017a.

Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International conference on machine learning*, pp. 11117–11128. PMLR, 2020a.

Ruohan Zhang, Calen Walshe, Zhuode Liu, Lin Guan, Karl Muller, Jake Whritner, Luxin Zhang, Mary Hayhoe, and Dana Ballard. Atari-HEAD: Atari human eye-tracking and demonstration dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6811–6820, apr 2020b. doi: 10.1609/aaai.v34i04.6161. URL `https://doi.org/10.1609%2Faaai.v34i04.6161`.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4352–4360, 2017b. doi: 10.1109/CVPR.2017.463.

Supplementary Material

Outline

The appendix is organized as follows:

- In Appendix A, we define the metrics we used to evaluate marginal and conditional probability estimation (Appendix A.1) and confidence interval estimation (Appendix A.2).
- In Appendix B, we provide additional information about the datasets utilized in this study.
- In Appendix C, we include additional details regarding the model and hyperparameter search.
- In Appendix D, we include additional results. Appendix D.1 reports results for marginal probability estimation with comprehensive metrics and more reliability diagrams. We also include comparisons with the ground truth probabilities for the synthetic FaceMed dataset. In Appendix D.2, we report results for conditional probability estimation. In Appendix D.3, we report results for confidence interval estimation.
- In Appendix E, we provide a sensitivity analysis of the effect of regularization applied at different entries in a sequence.
- In Appendix F, we present an example illustrating how the probabilities estimated by foCus change when we condition on a certain event.
- In Appendix G, we provide results on applying regularization on a transformer based simulator.
- In Appendix H, we show the evolution of the performance of foCus as we train the simulator with different types of regularization.

A Evaluation metrics

A.1 Marginal and conditional probability

For each class $a \in \{1, \cdots, c\}$ and each entry i of the sequence, we evaluate the estimated probabilities $\mathrm{P}(Y_i = a \,|\, X = x)$ or $\mathrm{P}(Y_i = a \,|\, Y_j = b, X = x)$ using the relevant data (for marginal probabilities, these are all sequences; for conditional probabilities, all sequences such that the jth entry equals b). The entry-level metrics we propose are defined below. The sequence-level metrics are computed by averaging the entry-level metrics across all entries.

Macro AUC. The Area Under the Curve (AUC) per class is computed separately for each class a at each entry i using a one-vs-all approach. We aggregate all the class AUCs via averaging to obtain the overall macro AUC.

Brier Score The Brier Score (BS) evaluates both calibration and discriminative ability. The Brier Score per class is the mean-squared error between the predicted probabilities and binarized label per class. The entry-level BS is the mean of BS per class, averaged over all the classes.

Cross Entropy The cross-entropy (CE) loss is computed following equation 3.

Expected Calibration Error. We use confidence expected calibration error (ECE) (Guo et al., 2017) to assess calibration. The *confidence* is defined as the predicted probability of the class a with the highest estimated probability. These confidences are grouped into B bins, based on B-quantiles. ECE is the mean absolute difference between the accuracy (empirical probability of correct predictions) and the average confidence within each bin. A lower ECE indicates better calibration.

To provide further insight, we also plot reliability diagrams. These diagrams compare the empirical probability (accuracy) with the estimated probability (confidence) in each bin. A well-calibrated model will produce a reliability diagram that is close to the diagonal.

A.2 Confidence intervals

Let the ground truth time-to-event for the kth data point be denoted as $T[k]$, and the estimated confidence interval as $I_\alpha[k] = \left[q_{(1-\alpha)/2}[k], q_{(1+\alpha)/2}[k]\right]$.

Coverage Probability The coverage probability measures the proportion of samples where the true time-to-event $T[k]$ lies within the estimated confidence interval $I_\alpha[k]$. This metric reflects how well calibrated the estimated confidence intervals are. Ideally, for a confidence level α, the coverage probability should equal α.

Relative Confidence Interval Width The width of the confidence interval quantifies the uncertainty in the model estimates. A wider confidence interval indicates higher uncertainty. To account for different sequence lengths across datasets, we normalize the confidence interval width by the mean of the true time-to-event averaged over each dataset. The relative confidence interval width is defined as:

$$\frac{\frac{1}{N}\sum_{k=1}^{N}\left(q_{(1+\alpha)/2}[k] - q_{(1-\alpha)/2}[k]\right)}{\frac{1}{N}\sum_{k=1}^{N} T[k]}, \tag{5}$$

where N is the number of data.

Relative Mean Absolute Error (MAE) The relative MAE is the mean of the absolute difference between the estimated and ground truth time-to-event, normalized by the mean of the ground truth times-to-event. We estimate the time-to-event by averaging over the time-to-event values $\widehat{T}^{(1)}$, ..., $\widehat{T}^{(m)}$ corresponding to the m Monte Carlo simulations:

$$\text{Relative MAE} = \frac{\sum_{k=1}^{N}\left|T[k] - \frac{1}{M}\sum_{m=1}^{M}\widehat{T}^{(m)}[k]\right|}{\sum_{k=1}^{N} T[k]}. \tag{6}$$

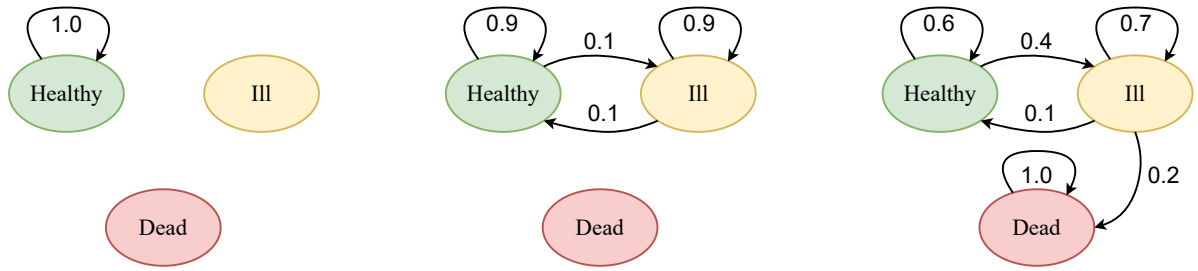
B Datasets

FaceMed FaceMed is a synthetic dataset based on the UTKFace dataset (Zhang et al., 2017b), which contains face images along with corresponding ages. We simulate health status transitions between three distinct states: 1 for healthy, 2 for ill, and 3 for dead. The simulated health states per each year form a sequence for each patient. The underlying transition probabilities among these health states are determined by the individual's age. The goal of the sequence prediction task is to forecast a patient's health status trajectory using their facial images.

To simulate the dynamics of health status, we use an age-dependent Markov process, where the health status at the ith entry, Y_i, only depends on the previous state Y_{i-1}, for any $i > 1$. The conditional probability between states is given by:

$$\mathrm{P}(Y_i = a \,|\, Y_{i-1} = b) = p_{b,a}. \tag{7}$$

The transitions among health statuses are illustrated in Figure 5. Every individual is healthy as the initial state. The transition probabilities for the simulation are defined as follows: For individuals younger than 40, the health status never changes (Figure 5 (a)). For those aged 40 to 80, the health status can transition between healthy and ill, with transition probabilities $p_{1,1} = p_{2,2} = 0.9$, $p_{1,2} = p_{2,1} = 0.1$ (Figure 5 (b)). For individuals older than 80, the likelihood of becoming ill increases, with transition probabilities $p_{0,\cdot} = (0.6, 0.4, 0)$; for ill individuals in this age group, there is a chance of death, reflected in the transition probabilities $p_{1,\cdot} = (0.1, 0.7, 0.2)$ (Figure 5 (c)). These probabilities vary as the individual becomes "older" in the simulation, reflecting the increasing risks associated with aging. The average survival time of the simulated sequences is 45.45 years. In the experiments, For individuals who die before 100 years from the beginning, their sequences are padded to cover 100 years. The dataset is split into training, validation, and test sets with 16641, 4738, and 2329 samples, respectively.



(a) $0 - 40$ years old (b) $40 - 80$ years old (c) > 80 years old

Figure 5: **Markov process used to simulate health-status transitions in FaceMed.**

Table 2: **Additional information about video game data.** The table summarizes each video game dataset's training set size, validation set size, test set size, average sequence length until the next scoring point, sequence length after padding, and action sampling frequency.

Game Name	Training	Validation	Testing	Ave. Length (seconds)	Padded Length	Sampling freq.
Seaquest	104,806	17,556	17,650	1.0071	200	3
River Raid	104,252	17,584	17,592	1.4082	300	2
Bank Heist	105,765	17,553	17,548	1.7645	300	2
H.E.R.O.	98,553	16,769	16,885	1.4586	300	2
Road Runner	239,428	68,332	34,177	2.5796	300	4

Atari games Our real-data experiments are based on the Atari-HEAD dataset (Zhang et al., 2020b), a large-scale, high-quality imitation learning dataset that captures human actions alongside eye movements and game frames while playing Atari video games. The dataset employs a unique semi-frame-by-frame gameplay format, where the game pauses at each frame until the player performs a keyboard action. This ensures that each frame in the video of game and the corresponding human action are aligned.

In this work, the sequence prediction task aims to predict a player's action trajectory based on a given game frame. Each frame serves as a high-dimensional input X. The subsequent actions until the next scoring event are treated as a sequence Y. Since actions are recorded at a high frame-by-frame frequency, they often repeat several times before transitioning to a new action, yielding sequences with redundant information. To reduce this redundancy, we sample actions at a constant frequency determined by the number of frames per sequence entry for each game. The corresponding time t in the game at the ith entry of the sequence can be recovered from the entry as follows: $t(\text{second}) = \text{sampling freq.}/60(\text{Hz}) \times i$. We pad the sequences with an *end-of-game* value to ensure that all sequences have the same length.

Experiments are conducted on five games from Atari-HEAD: **Seaquest**, **River Raid**, **Bank Heist**, **H.E.R.O.**, and **Road Runner**. They represent a broad category of video games available in Atari-HEAD. We use 70 percent of gameplay as training set, 20 percent as the validation set, and 10 percent as the test set. See Table 2 for more detailed information about the data corresponding to each game.

C Technical Details

C.1 Model Architecture

As illustrated in Figure 2 (a), we employ a neural-network simulator. The simulator consists of a convolutional neural network (CNN) that encodes the input image x, and a recurrent neural network (RNN). The CNN encoder is a Resnet-18 (He et al., 2015), which produces an image embedding of dimension 256. The RNN decoder is implemented as a single-layer LSTM with a hidden vector size 256. The image embedding

from the CNN encoder is fed to the RNN decoder as the initial hidden vector h_0. The RNN iteratively updates its hidden state h_i based on the previous hidden state h_{i-1} and the preceding input y_{i-1} from the $(i-1)$th entry of the sequence. For each step $i > 1$, the RNN outputs a logit vector $\mathbf{z}(x, y_1, \ldots, y_{i-1}) \in \mathbb{R}^c$ through a linear layer with input h_i. The logit vector is normalized with a softmax function and used to estimate the class probabilities of a multinoulli distribution:

$$p_\theta\left(a \,|\, x, y_1, ..., y_{i-1}\right) = \frac{\exp\left(\mathbf{z}(x, y_1, ..., y_{i-1})[a]\right)}{\sum_{k=1}^{c} \exp(\mathbf{z}\left(x, y_1, ..., y_{i-1}\right)[k])}, \tag{8}$$

where $a \in \{1, \ldots, c\}$ and $\mathbf{z}(x, y_1, ..., y_{i-1})[k]$ is the kth entry of the logit vector.

For the health-status prediction task, the RNN decoder outputs a 3-dimensional logit vector corresponding to the three possible health states ($c = 3$). In the case of Atari games, where each action can belong to one of 19 possible classes, the RNN generates a 19-dimensional logit vector ($c = 19$).

C.2 Training

As explained in Section 6, the neural network simulator is trained by minimizing the cross-entropy loss between the predicted distribution and the one-hot encoded ground truth for each variable, with an additional regularization term that penalizes the ℓ_2-norm of each entry in the logit vector.

We train each model for 200 epochs for each scenario, with a batch size of 256, using the Adam optimizer without weight decay. The learning rates are kept constant for each scenario: 1×10^{-5} for Seaquest, River Raid, Bank Heist, and Road Runner, and 5×10^{-5} for H.E.R.O.

Model selection during training is challenging due to the numerous metrics involved in probability estimation tasks. Figures 13, 14, and 15 show the evolution of different metrics during training. We observe that the models are most discriminative (lower relative MAE and higher AUC) toward the end of training in most scenarios.

C.3 Inference

Figure 2(b) illustrates how the neural-network simulator is used to obtain a sample sequence $(\hat{y}_1, \ldots, \hat{y}_\ell)$. The input image x is fed into the CNN encoder, producing a hidden vector h_0 that is fed into the RNN decoder to then generate the simulated sequence iteratively. At each iteration $i \in \{1, ..., \ell\}$, the input of the RNN decoder is the value $\hat{y}_{i-1}$ of the previous entry (except for $i = 1$) and the hidden vector h_{i-1}. The outputs are an estimate of the conditional distribution of Y_i given the previous entries and $X = x$, and an updated hidden vector h_i. The ith entry $\hat{y}_i$ of the sample sequence is sampled from this conditional distribution. Since each entry is drawn randomly from the predicted distribution, the simulator is capable of generating multiple different sequences from the same input image, acting as a simulator. When performing Monte Carlo estimation, we generate $m = 100$ sampled sequences for each input image.

C.4 Hyperparameter search

The hyperparameters for time-dependent regularization were determined via the following procedure:

1. For $1 \leq i \leq k_1$ (where k_1 is a hyperparameter) we use the sequence-level ECE of marginal probabilities (see Section 4.1) computed over validation set to iteratively select λ_i, setting $\lambda_j = 0$ for all $j > i$.
2. For $k_1 < i \leq k_2$ (where k_2 is a hyperparameter) we constrain all the parameters to equal the same constant, $\lambda_i = \lambda_{\text{all}}$, selected also based on the validation ECE.
3. For $i > k_2$ we set $\lambda_i = 0$.

We set $k_1 = 3$. To determine each λ_i and λ_{all} we performed a search on the fixed grid $\{0.001, 0.005, 0.01, 0.05\}$ based on the validation ECE for the marginal probability estimation task. For k_2 we used the grid $\{1, 11, 21, 51, 101\}$.

Table 3: **Chosen regularization hyperparameters.** This table shows the chosen regularization parameters in all scenarios for both time-dependent and constant regularization.

Scenarios	Time-dependent λ's	Constant λ's
Seaquest	$\lambda_{1:3} = 0.05, 0.01, 0.05 \quad \lambda_{4:200} = 0$	$\lambda_{1:200} = 0.001$
River Raid	$\lambda_{1:6} = 0.01 \quad \lambda_{7:300} = 0$	$\lambda_{1:300} = 0.001$
Bank Heist	$\lambda_1 = 0.05 \quad \lambda_{2:11} = 0.01 \quad \lambda_{12:300} = 0$	$\lambda_{1:300} = 0.001$
H.E.R.O.	$\lambda_1 = 0.01 \quad \lambda_{2:6} = 0.005 \quad \lambda_{7:300} = 0$	$\lambda_{1:300} = 0.001$
Road Runner	$\lambda_1 = 0.01 \quad \lambda_{2:21} = 0.005 \quad \lambda_{22:300} = 0$	$\lambda_{1:300} = 0.001$
FaceMed	$\lambda_{1:3} = 0.01 \quad \lambda_{4:5} = 0.005 \quad \lambda_{5:50} = 0.001$	$\lambda_{1:300} = 0.001$

For constant regularization, we constrain all the parameters to be the same, $\lambda_i = \lambda_{\text{const}}$. Then λ_{const} we performed a search on the grid $\{0.001, 0.005, 0.01, 0.05\}$, also based on validation ECE. The hyperparameters chosen for both regularization methods are listed in Table 3.

D Supplementary Experimental Results

D.1 Marginal Probability

Figure 6 shows plots of the entry-level metrics (ECE, AUC, BS, and CE) for marginal probability estimation, complementing Figure 3. Time-dependent regularization leads to a substantial improvement in calibration, as demonstrated by the significantly lower ECE of the time-regularized model, which has a comparable AUC to the model without regularization. As a result, the cross-entropy and Brier Score metrics are also improved. Constant regularization also improves the probability estimates, but not as much as time-dependent regularization. Figure 7 shows additional reliability diagrams like the ones in Figure 3 and includes a comparison with constant regularization, confirming that time-dependent regularization consistently improves calibration for individual entries.

In Section 6.1, we also compare the estimated marginal probability with underlying data generating distribution on the synthetic FaceMed data. Since the ground truth sequences in FaceMed are generated using an assumed transitional probability model, as detailed in Appendix B, we can analytically compute the *ground truth* marginal probability of Y_i based on the transitional probabilities and the marginal probability of the previous entry, Y_{i-1}:

$$\mathrm{P}(Y_i = a \mid X = x) = \sum_{b=1}^{c} \mathrm{P}(Y_{i-1} = b \mid X = x)\, \mathrm{P}(Y_i = a \mid Y_{i-1} = b) \tag{9}$$

Starting from an initial state of "healthy," we compute the ground truth marginal probabilities for each entry in the sequence and compare them against the estimates obtained via foCus. We calculate the root mean square error (RMSE) between the ground truth and the estimated marginal probabilities for each entry, then aggregate these values to derive a sequence-level RMSE by averaging. The unregularized baseline method and constant regularization yield RMSE of 0.1847 ± 0.0014 and 0.1754 ± 0.0028, respectively, while time-dependent regularization reduces the RMSE to 0.1720 ± 0.0009. This result further demonstrates the improved performance achieved by time-dependent regularization.

D.2 Conditional Proabability

We evaluate the conditional probability estimation, given a fixed event in each scenario. For conditional probability estimation, FaceMed is conditioned on the status of the first year being Healthy. Video games are conditioned on the first entry being equal to the most frequent first action in the training set. Specifically, Seaquest is conditioned on first action as *NOOP* (no operation), River Raid as *NOOP*, Bank Heist as *Right*, H.E.R.O. as *NOOP*, and Road Runner as *Left*. Table 4 reports the sequence-level metrics for conditional probability estimation. Figure 8 shows entry-level metrics (ECE, AUC, BS, and CE) for conditional prob-

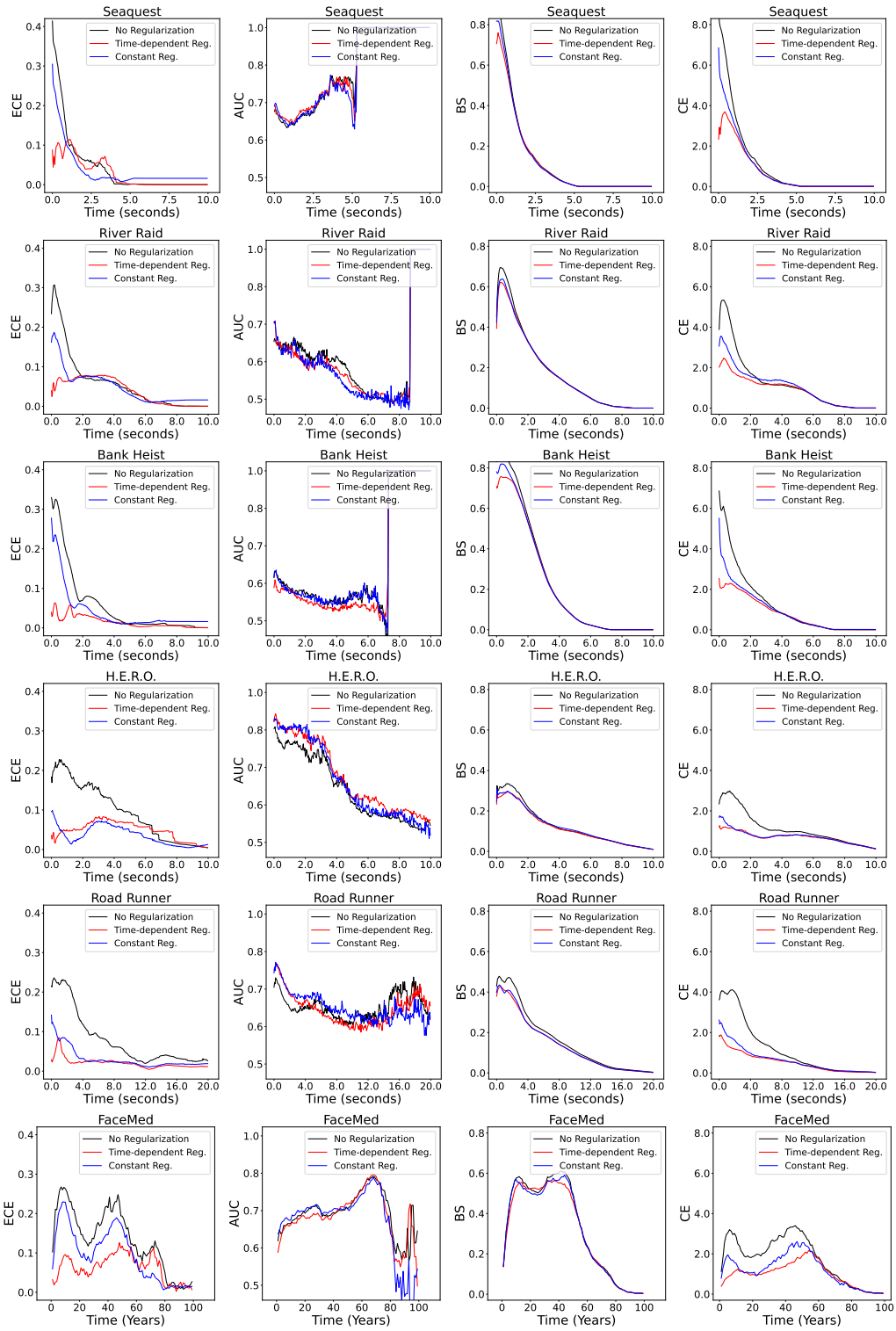


Figure 6: **Entry-wise metrics for marginal probability estimation.** The figure plots the entry-level ECE, AUC, BS, and CE of the proposed foCus framework for estimation of marginal probabilities. It compares versions of foCus: without regularization (black), time-dependent regularization (red), and constant regularization (blue). Both constant regularization and time-dependent regularization improve calibration and overall estimation quality compared to without regularization. Time-dependent regularization's improvement is more significant than constant regularization.

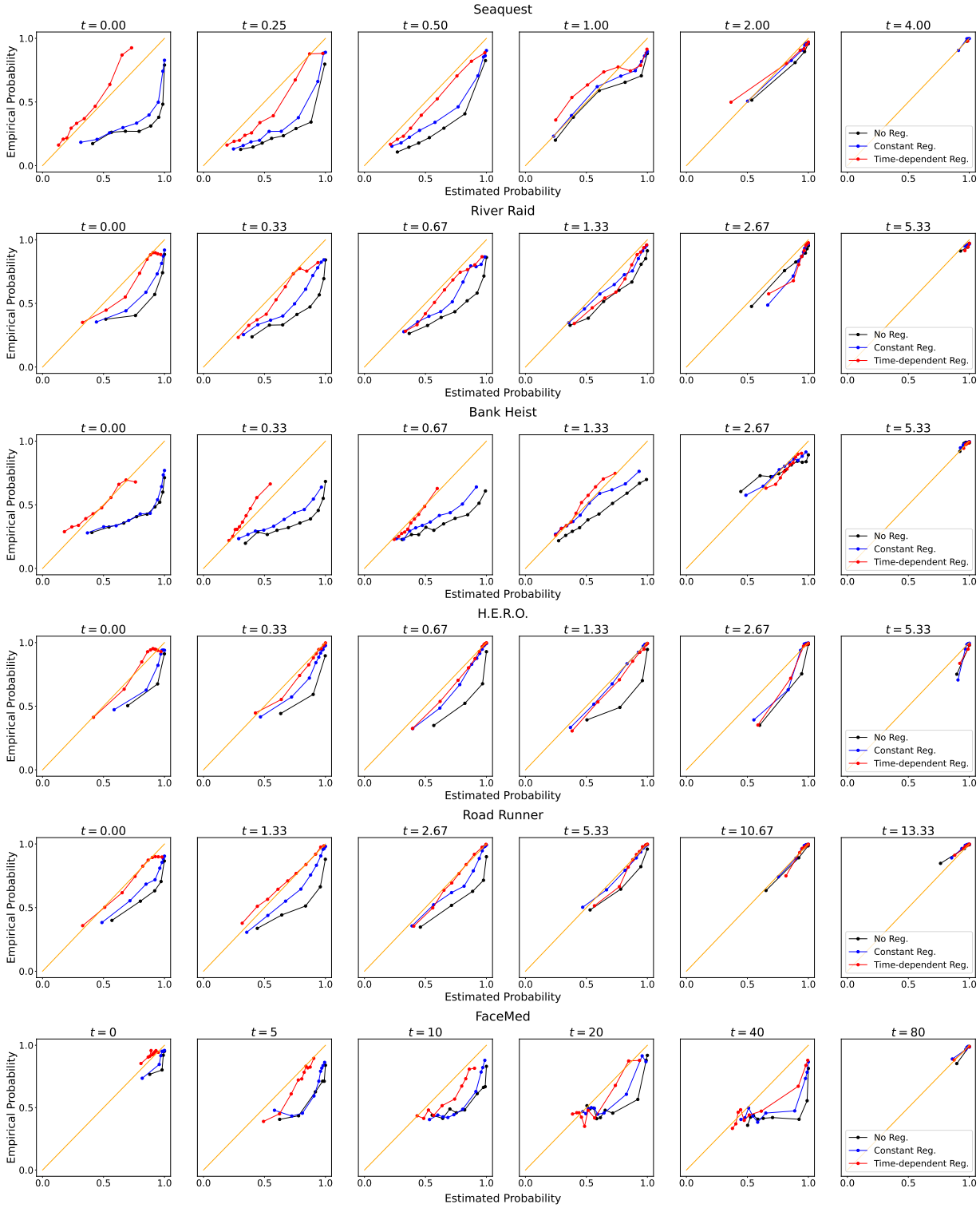


Figure 7: **Entry-wise reliability diagrams for marginal probability estimation.** The figure, supplementing Figure 3 presents reliability diagrams at additional entries in the sequence. Each diagram compares three versions of the proposed foCus framework: without regularization (black), time-dependent regularization (red), and constant regularization (blue). The plots demonstrate that both regularization methods improve calibration at each sequence entry, with time-dependent regularization showing the most substantial improvements.

Table 4: **Conditional probability estimation.** The table, discussed in Section 6.1, reports sequence-level metrics evaluating the performance of the proposed foCus framework for estimation of conditional probabilities (see Section 3). We compare versions of foCus without regularization (see Section 5) and with constant and time-dependent regularization (see Section 6). Results are presented as mean ± standard error from three independent model realizations. Time-dependent regularization improves calibration substantially (lower ECE), while maintaining a comparable AUC, which results in superior probability estimates (lower CE and BS).

Scenario	Regularization	ECE (↓)	AUC (↑)	CE (↓)	BS (↓)
Seaquest	✗	0.0590 ± 0.0035	0.8887 ± 0.0045	0.9032 ± 0.0779	0.1386 ± 0.0016
	time-dependent	0.0316 ± 0.0017	**0.8938 ± 0.0009**	**0.6070 ± 0.0110**	0.1301 ± 0.0053
	constant	**0.0298 ± 0.0017**	0.8827 ± 0.0021	0.6227 ± 0.0278	**0.1289 ± 0.0007**
River Raid	✗	0.0841 ± 0.0008	**0.7007 ± 0.0046**	1.4968 ± 0.0497	0.2264 ± 0.0031
	time-dependent	**0.0652 ± 0.0012**	0.6997 ± 0.0036	**1.1587 ± 0.0117**	0.2248 ± 0.0018
	constant	0.0689 ± 0.0005	0.6926 ± 0.0026	1.3235 ± 0.0434	**0.2225 ± 0.0015**
Bank Heist	✗	0.0534 ± 0.0035	**0.7254 ± 0.0036**	1.0618 ± 0.0563	0.2287 ± 0.0030
	time-dependent	**0.0184 ± 0.0005**	0.7033 ± 0.0021	**0.7616 ± 0.0046**	**0.2141 ± 0.0008**
	constant	0.0405 ± 0.0022	0.7149 ± 0.0124	0.8431 ± 0.0080	0.2168 ± 0.0013
H.E.R.O.	✗	0.0993 ± 0.0039	0.7063 ± 0.0002	1.0290 ± 0.0371	0.1195 ± 0.0021
	time-dependent	0.0645 ± 0.0087	**0.7420 ± 0.0132**	**0.7189 ± 0.0289**	**0.1151 ± 0.0029**
	constant	**0.0418 ± 0.0019**	0.7249 ± 0.0141	0.7283 ± 0.0572	0.1208 ± 0.0024
Road Runner	✗	0.0870 ± 0.0061	0.6790 ± 0.0091	1.2813 ± 0.0244	0.1693 ± 0.0039
	time-dependent	**0.0207 ± 0.0012**	0.6772 ± 0.0107	**0.5489 ± 0.0089**	**0.1449 ± 0.0003**
	constant	0.0321 ± 0.0032	**0.6905 ± 0.0135**	0.6654 ± 0.0304	0.1472 ± 0.0012
FaceMed	✗	0.1488 ± 0.0026	0.7533 ± 0.0066	1.6428 ± 0.0147	0.3465 ± 0.0012
	time-dependent	**0.0888 ± 0.0065**	0.7603 ± 0.0040	**1.0224 ± 0.0348**	**0.3350 ± 0.0018**
	constant	0.0984 ± 0.0042	**0.7652 ± 0.0050**	0.9875 ± 0.0372	0.3354 ± 0.0016

ability estimation. Similar to the case of marginal probability estimation Appendix D.1, time-dependent regularization improves calibration and the overall quality of probability estimation.

D.3 Confidence Interval for Time-to-event

Table 5 reports the results for time-to-event confident interval estimation. In this case, we observe a certain trade-off between discriminative performance, quantified by the relative MAE, and calibration, quantified by coverage probabilities. The MAE of no regularization and constant regularization are consistently lower than those of time-dependent regularization, but the coverage probabilities of time-dependent regularization are a lot closer to 90% (between 69% and 92%, compared to at most 70% for the other two methods).

Figure 9, complementing Figure 4, presents heatmaps of the confidence interval widths for different ground-truth time-to-event (upper panel of each subplot) for River Raid, H.E.R.O., Road Runner, and Bank Heist, as well as a histogram showing the fraction of intervals containing the ground-truth (lower panel of each subplot). Unregularized foCus produces very narrow confidence intervals with very poor coverage, whereas time-dependent regularization yields intervals that tend to be larger when the true time-to-event becomes greater (and hence generally more uncertain), achieving much better coverage. These finding further supports that time-dependent regularization performs better in confidence interval estimation.

E Regularization Sensitivity Analysis

In this section, we explore how regularization at a single entry affects model performance. Using the Seaquest game, we train four versions of foCus, each applying regularization to a different entry in the sequence, cor-

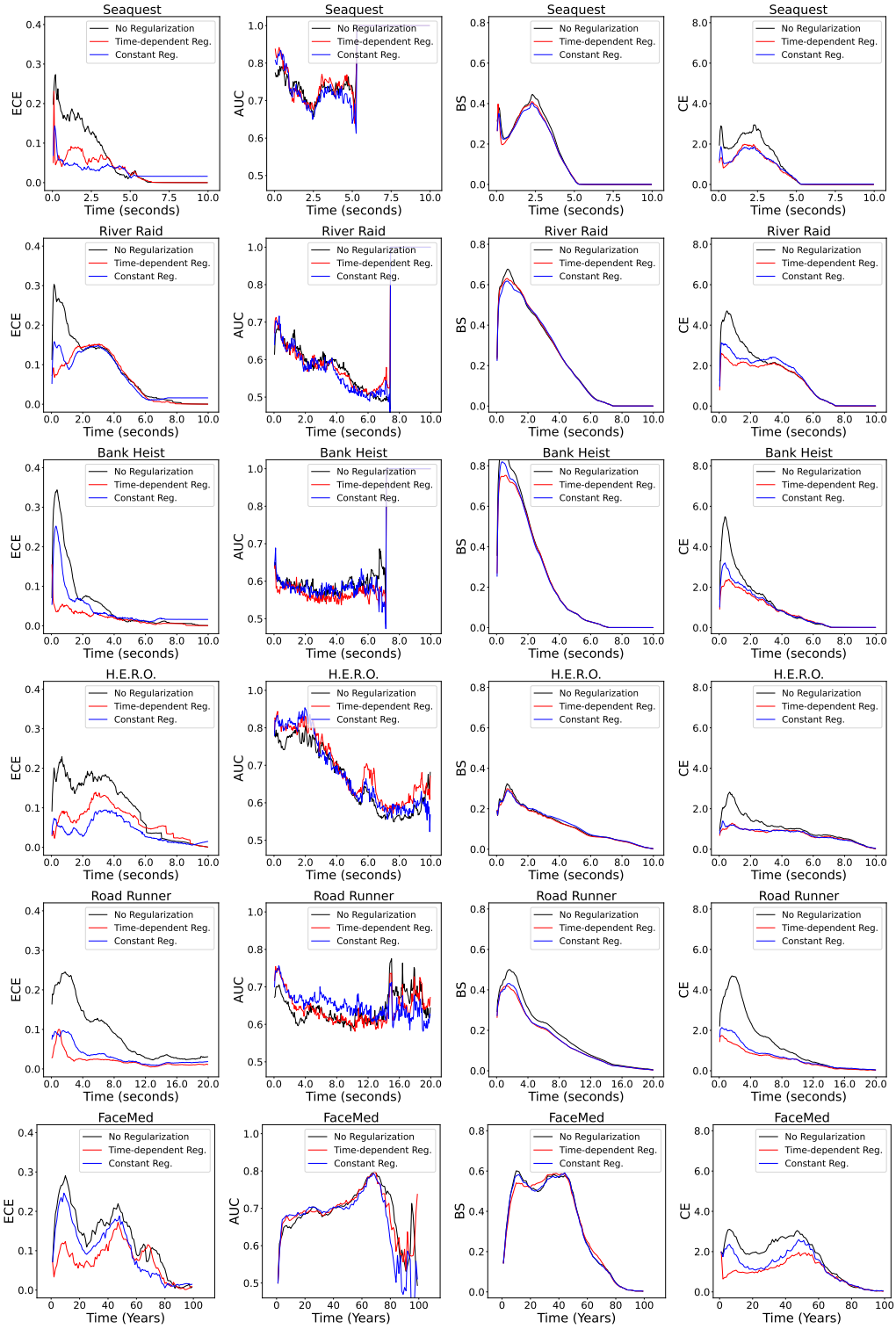


Figure 8: **Entry-wise metrics for conditional probability estimation.** The figure plots the entry-level ECE, AUC, BS, and CE of the proposed foCus framework for estimation of conditional probabilities. It compares versions of foCus: without regularization (black), time-dependent regularization (red), and constant regularization (blue). Both constant regularization and time-dependent regularization improve calibration and overall estimation quality compared to without regularization. Time-dependent regularization's improvement is more significant than constant regularization.

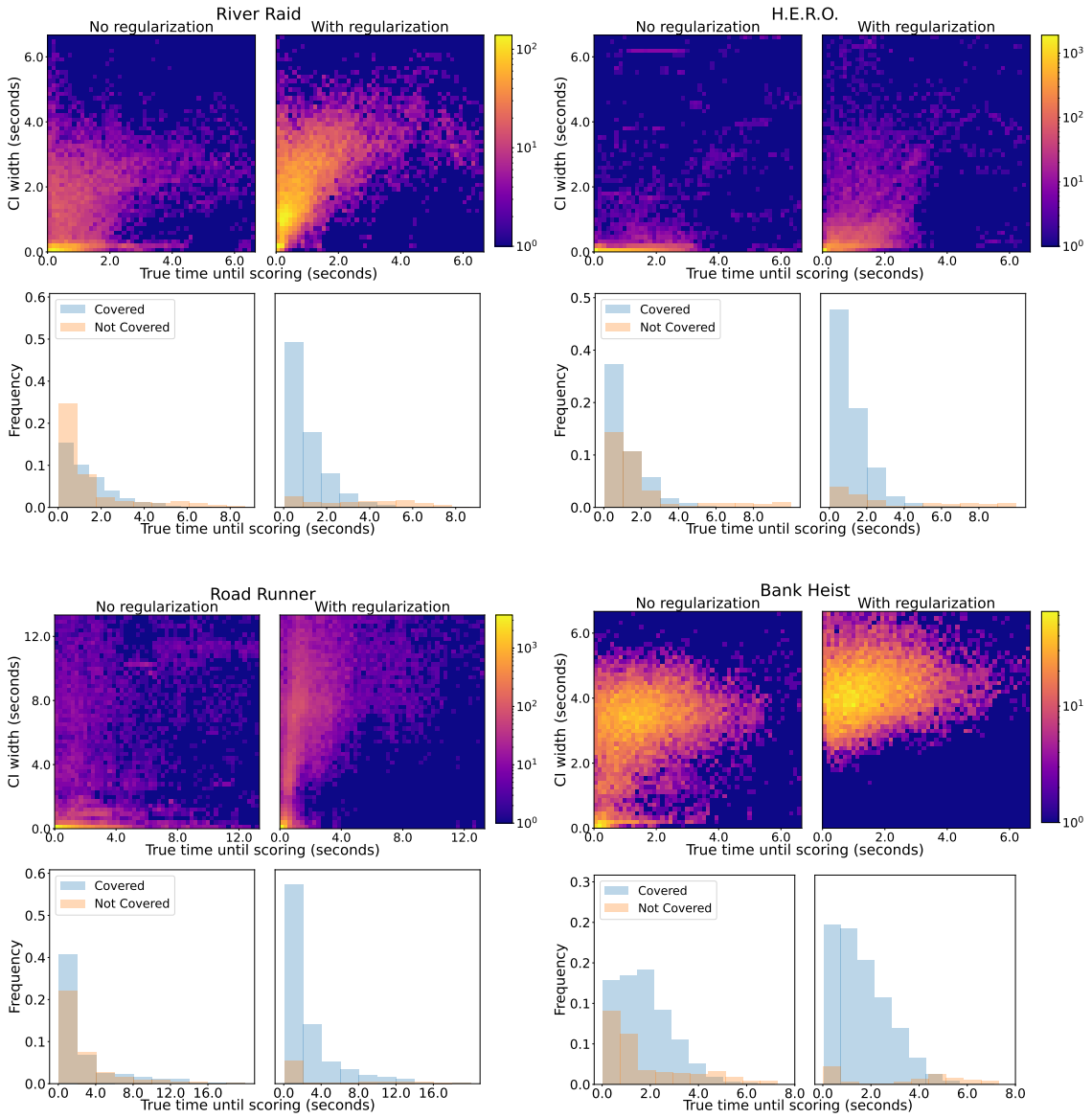


Figure 9: **Confidence intervals for time-to-event prediction and coverage probability.** The upper panel of each subplot shows heatmaps of the length of 0.9 confidence intervals for time-to-event prediction using the proposed foCus framework without and with time-dependent regularization. The histograms below show the frequency of intervals containing the true times, as a function of the true times. Unregularized foCus produces short intervals with poor coverage, whereas regularization yields intervals that tend to be larger when the ground-truth times are larger, and are much better calibrated. The subplots correspond to the River Raid, H.E.R.O., Road Runner, and Bank Heist datasets.

Table 5: **Performance comparison of time-to-event prediction confidence intervals (CI) across six scenarios.** The table presents metrics from the same experiments as in Table 1. The time-dependent regularization model achieves significantly better 90% CI ($I_{0.9}$) coverage, indicating improved calibration of uncertainty estimation.

Scenario	Regularization	Coverage Prob. of $I_{0.9}$	Relative Width of $I_{0.9}$	Relative MAE (↓)
Seaquest	✗	0.3420 ± 0.0177	0.7401 ± 0.0149	0.4622 ± 0.0112
	time-dependent	**0.7065 ± 0.0121**	2.1718 ± 0.0521	0.5297 ± 0.0079
	constant	0.4440 ± 0.0162	0.9409 ± 0.0266	**0.4538 ± 0.0069**
River Raid	✗	0.4463 ± 0.0105	1.0734 ± 0.0398	0.5679 ± 0.0044
	time-dependent	**0.8345 ± 0.0090**	1.8720 ± 0.0520	0.5674 ± 0.0037
	constant	0.6241 ± 0.0018	1.2079 ± 0.0241	**0.5580 ± 0.0012**
Bank Heist	✗	0.7066 ± 0.0134	1.8289 ± 0.0330	0.5811 ± 0.0052
	time-dependent	**0.9243 ± 0.0048**	2.8099 ± 0.0535	0.5991 ± 0.0080
	constant	0.8133 ± 0.0083	1.9123 ± 0.0335	**0.5464 ± 0.0025**
H.E.R.O.	✗	0.2252 ± 0.0009	0.3712 ± 0.0122	**0.2849 ± 0.0057**
	time-dependent	**0.6850 ± 0.0121**	0.9035 ± 0.0266	0.3316 ± 0.0041
	constant	0.5827 ± 0.0102	0.7158 ± 0.0330	0.3337 ± 0.0025
Road Runner	✗	0.2646 ± 0.0128	1.2432 ± 0.1125	0.6079 ± 0.0277
	time-dependent	**0.7507 ± 0.0086**	2.9315 ± 0.0277	0.6419 ± 0.0012
	constant	0.4971 ± 0.0073	1.6267 ± 0.1351	**0.5652 ± 0.0078**
FaceMed	✗	0.2789 ± 0.0039	0.1329 ± 0.0035	**0.1594 ± 0.0005**
	time-dependent	**0.7169 ± 0.0322**	1.0421 ± 0.0232	0.2311 ± 0.0066
	constant	0.5897 ± 0.0227	0.7632 ± 0.0212	0.4753 ± 0.0032

responding to 0, 3, 6, and 9 seconds. As shown in Tables 6 and 7, the model with regularization applied at the beginning (0 seconds) exhibits the most significant improvement in calibration, both for marginal probability estimation and time-to-event prediction confidence intervals. This suggests that applying regularization early in the sequence is crucial for maintaining proper calibration throughout the whole sequence. These results also validate our heuristic procedure to implement time-dependent regularization.

Table 6: **Marginal probability estimation performance sensitivity analysis on Seaquest.** We compare versions of foCus where regularization is applied to a single random variable which is 0, 3, 6, and 9 second(s) after the start of the sequence. Results are presented as mean ± standard error from three independent model realizations. Best performance is achieved when applying regularization to the random variable immediately after the sequence starts.

Reg. Time	ECE (↓)	AUC (↑)	CE (↓)	BS (↓)
0 second	**0.0290 ± 0.0008**	0.8748 ± 0.0033	**0.8073 ± 0.0120**	**0.1164 ± 0.0008**
3 seconds	0.0424 ± 0.0019	0.8705 ± 0.0027	1.0227 ± 0.0150	0.1227 ± 0.0011
6 seconds	0.0424 ± 0.0016	0.8783 ± 0.0024	0.9599 ± 0.0208	0.1211 ± 0.0009
9 seconds	0.0409 ± 0.0020	**0.8784 ± 0.0023**	0.9450 ± 0.0065	0.1212 ± 0.0013

F Conditional Probability Estimation

In this section, we illustrate the conditional probability estimation task by showcasing how early actions substantially influence probability predictions at later stages, as shown in Figure 10.

Table 7: **Time-to-event prediction confidence interval performance sensitivity analysis on Seaquest.** The table presents metrics on time-to-event prediction confidence intervals from the same experiments as in Table 6. Regularizing he random variable immediately after the sequence starts significantly improves 90% CI ($I_{0.9}$) coverage.

Reg. Time	Coverage Prob. of $I_{0.9}$	Relative Width of $I_{0.9}$	Relative MAE (↓)
0 second	**0.4972 ± 0.0208**	1.6983 ± 0.0497	0.5162 ± 0.0125
3 seconds	0.3577 ± 0.0171	0.7056 ± 0.0320	**0.4466 ± 0.0036**
6 seconds	0.3556 ± 0.0099	0.7338 ± 0.0126	0.4495 ± 0.0070
9 seconds	0.3578 ± 0.0016	0.7421 ± 0.0308	0.4522 ± 0.0138

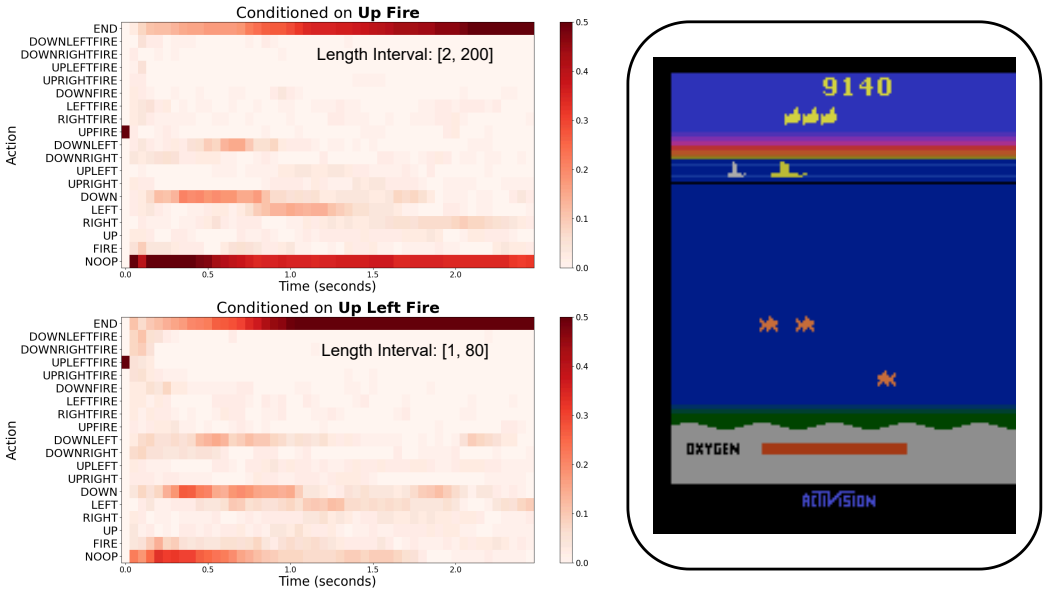


Figure 10: **Action distribution conditioned on different first actions.** The figure shows empirical action distribution for a Seaquest game frame (shown on the right) given first condition actions: Up Fire and action Up Left Fire.

We analyze the probabilities estimated from a frame of the game Seaquest (right panel of Figure 10) by foCus using time-dependent regularization. We estimate the conditional probabilities given two different first-step actions: *Up Fire* and *Up Left Fire*. The left panel of Figure 10 shows the resulting conditional probability estimation. When the first action is *Up Left Fire* instead of *Up Fire*, the frequency of *NOOP* (no operation) decreases significantly, accelerating the player's progress toward scoring. This makes intuitive sense, as *Up Left Fire* moves the green submarine closer to the enemy blue submarine, allowing the torpedo to reach its target more quickly. This analysis illustrates that the model effectively learns meaningful conditional probability estimates.

G Experiments with Transformer

In this section, we explore whether our findings generalize to different model architectures. We replaced the RNN component within the original model with a Transformer decoder; and applied this new model architecture to Seaquest and FaceMed. Similar to the RNN based model, the Transformer based model also significantly benefits from the step-wise regularization. Figure 11 shows marginal probability calibration

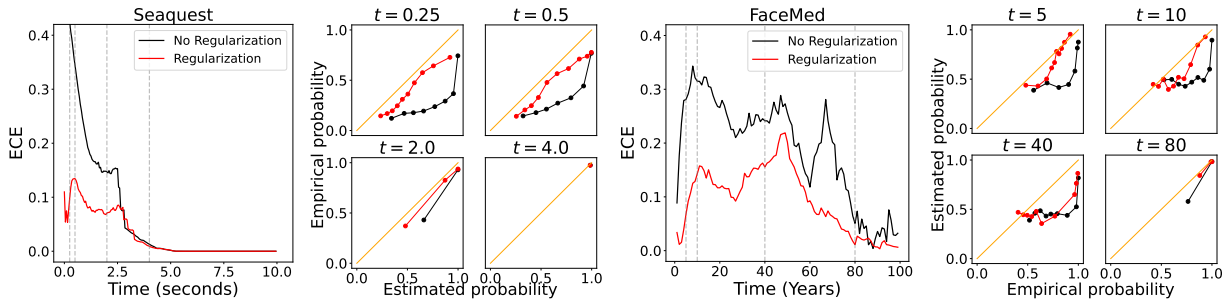


Figure 11: **Entry-wise calibration error and reliability diagrams for marginal probability estimation.** Similar to Figure 3, except the RNN component within the original model is replaced with a Transformer decoder. The results remain consistent, indicating that the findings hold across different sequence model architectures.

improves with regularization applied. Regularization can also help improve time-to-event calibration for the Transformer based model as in Figure 12.

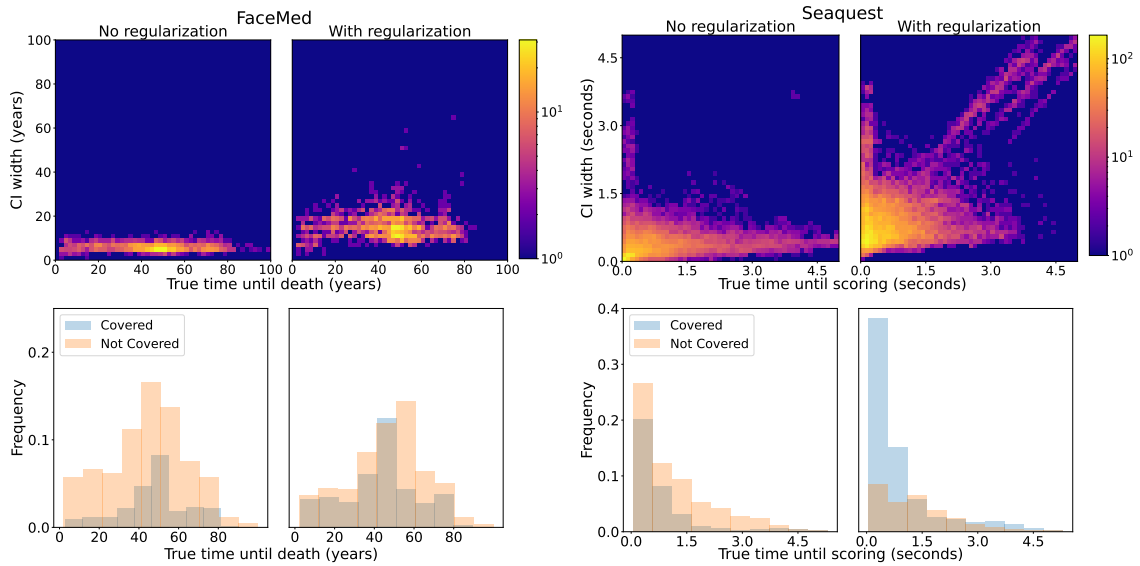


Figure 12: **Confidence intervals for time-to-event prediction and coverage probability.** Similar to Figure 4 but the RNN component within the original model is replaced with a Transformer decoder.

H Supplementary Figures

This section displays the learning curves for various sequence-level metrics on the test set at different epochs during training. All models were trained for 200 epochs, and the metrics were calculated separately for time-to-event prediction confidence intervals Figure 13, marginal probability estimation Figure 14, and conditional probability estimation Figure 15. These curves highlight the trade-offs between discriminability and calibration over the course of training: lower relative MAE is associated with lower coverage probabilities, and higher AUC tends to come with higher ECE. Despite such trade-offs, the figures demonstrate that among the three foCus variants, the model with time-dependent regularization strikes the best balance, maintaining calibration while improving discriminability throughout training.

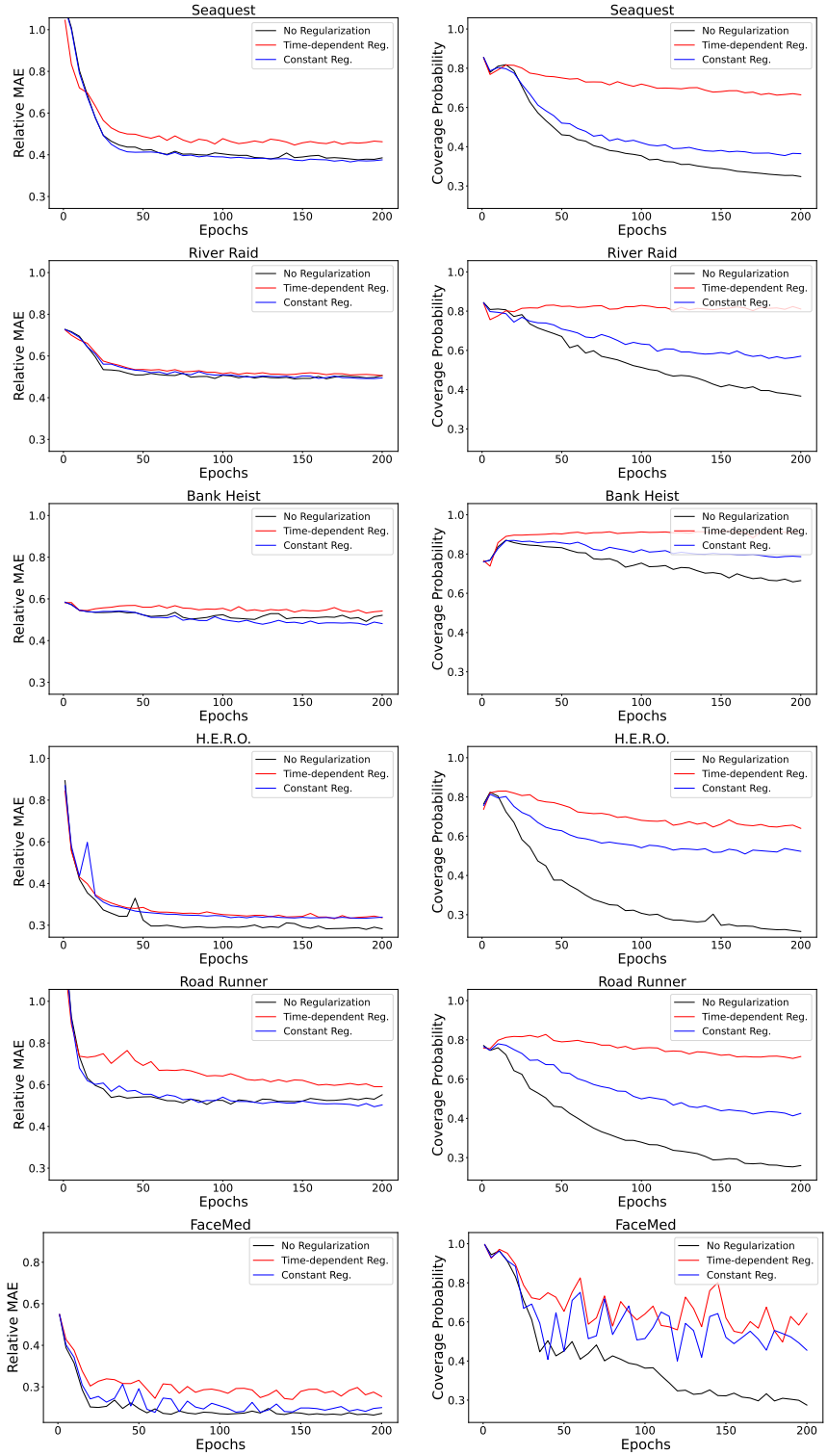


Figure 13: **Time evolution of metrics for time-to-event confident intervals.** It plots how coverage probability and relative MAE of $I_{0.9}$ evolve along training epochs for three versions of foCus: without regularization (black), time-dependent regularization (red), and constant regularization (blue). The time-dependent regularization model achieves significantly better coverage probability.

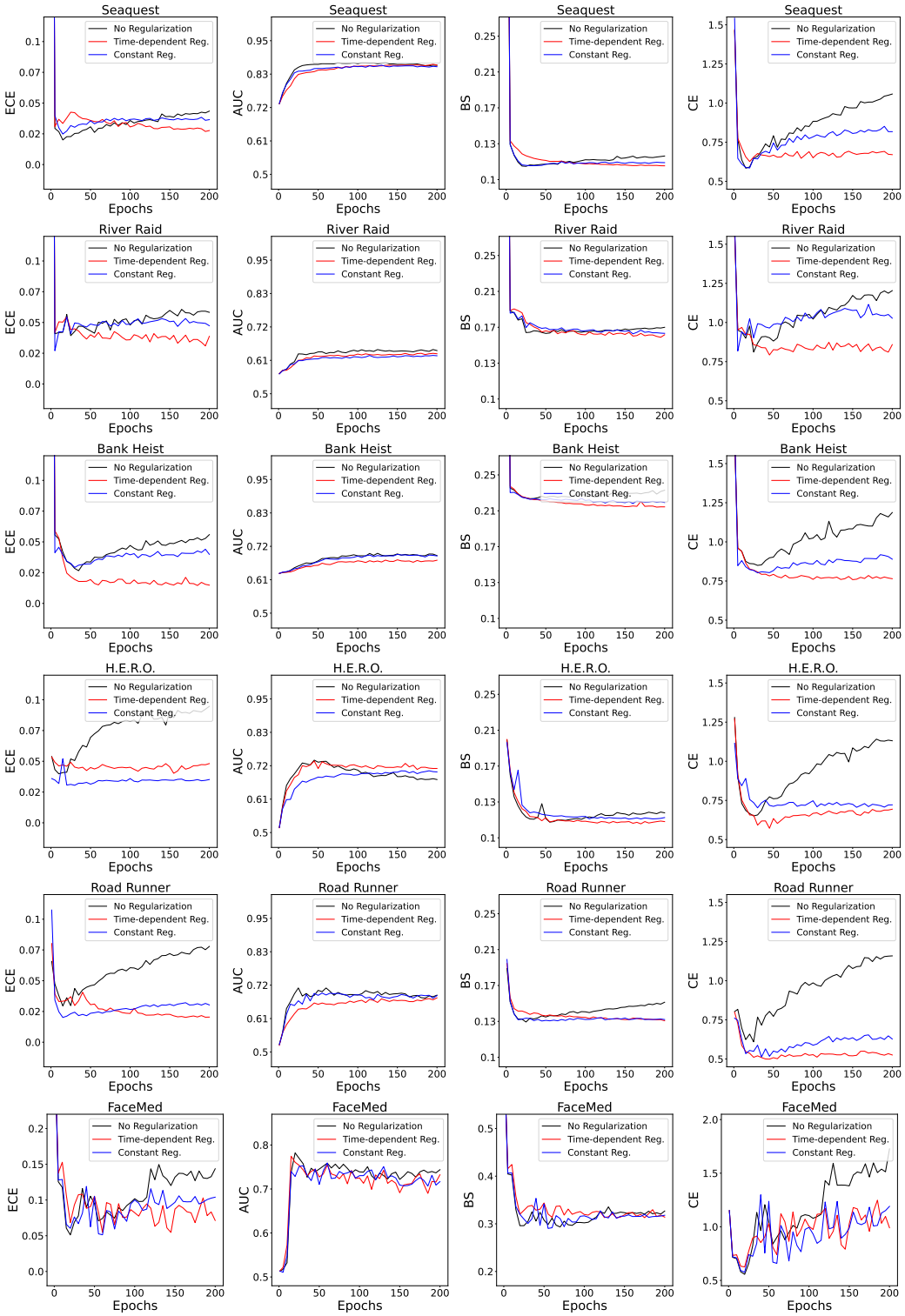


Figure 14: **Learning curves of sequence-level metrics for marginal probability estimation.** It plots how ECE, AUC, BS, and CE evolve along training epochs for three versions of foCus: without regularization (black), time-dependent regularization (red), and constant regularization (blue). The discriminability improves as the calibration decays. The time-dependent regularization model is able to keep calibrated while improving the discriminability.

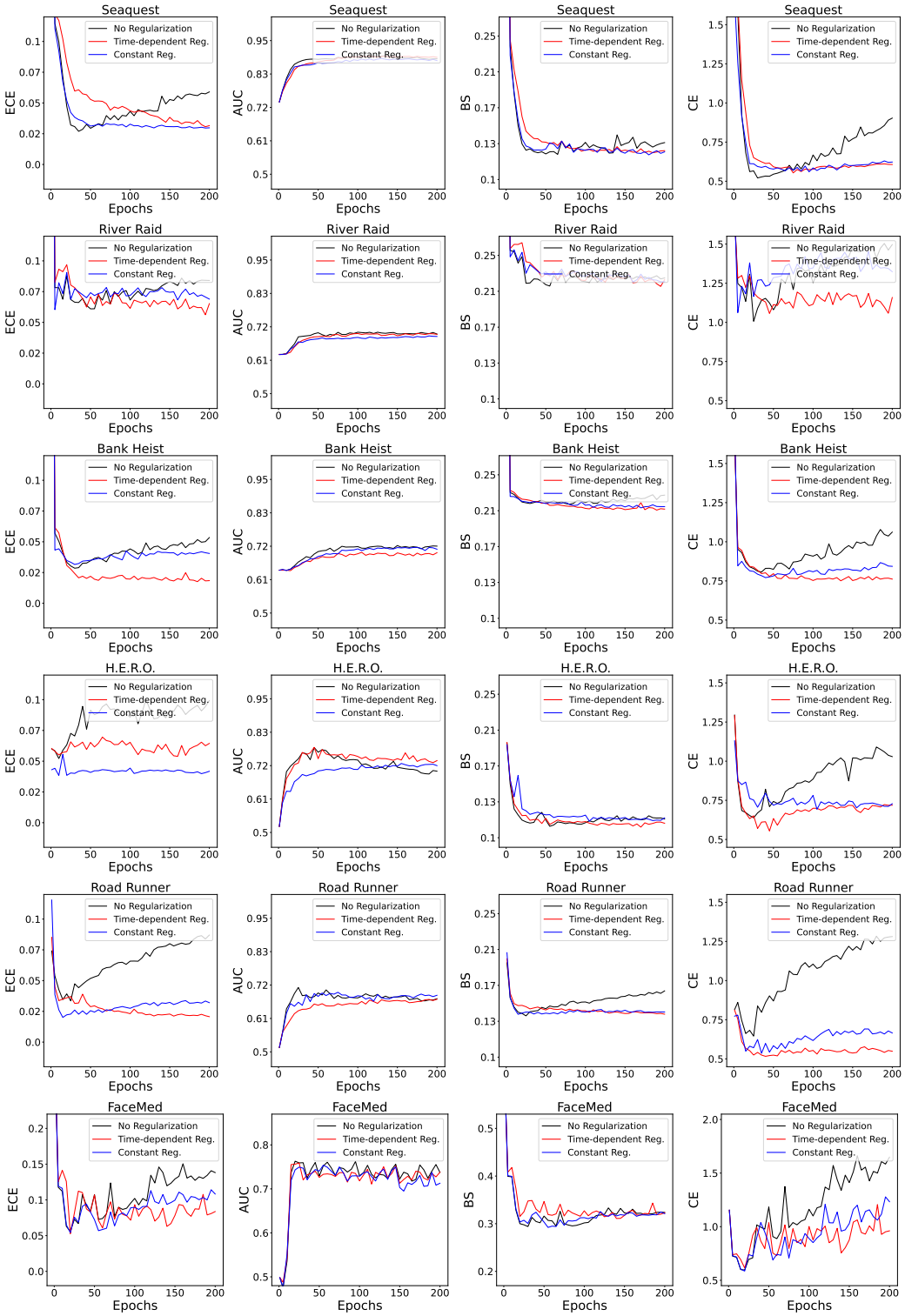


Figure 15: **Learning curves of sequence-level metrics for conditional probability estimation.** A similar pattern in Figure 14 is observed for conditional probability estimation from how ECE, AUC, BS, and CE evolve along training epochs for three versions of foCus.