MTL-MHOL: Predicting Online Conversions under Delayed Feedback and Data Sparsity

Anonymous authors
Paper under double-blind review

Abstract

This paper proposes the model-agnostic Multi-Task Learning Multi-Head Online Learning (MTL-MHOL) framework for conversion rate (CVR) prediction. Existing approaches typically address key challenges in CVR prediction, such as delayed feedback and data sparsity, in isolation or lack flexibility and practical applicability. MTL-MHOL adopts a time bucketing approach to account for delayed feedback and combines it with multi-task learning of an auxiliary task to mitigate data sparsity. The model is evaluated on datasets provided by a private company as well as on a dataset from Criteo, where it outperforms all benchmark models in terms of Negative Log Loss (NLL) and Relative Cross Entropy (RCE), correctly captures temporal trends in the data, and demonstrates strong performance when combined with either MLP or DeepFM. In particular, MTL-MHOL achieves up to 87% lift in RCE compared to the best performing benchmark. The code is freely available at https://anonymous.4open.science/r/MTL-MHOL.

Keywords: CVR Prediction, Delayed Feedback, Data Sparsity, Multi-Task Learning, Multi-Head Online Learning, MTL-MHOL

1 Introduction

In online advertising, conversion rate (CVR) prediction is central to modeling user behavior. It is defined as estimating the probability that a user who interacts with a company's online content performs a desired action, such as making a click or a purchase. A wide range of models have been developed to address three recurring challenges: delayed feedback, data sparsity and selection bias (Chapelle, 2014; Ktena et al., 2019; Yang et al., 2021; Gao & Yang, 2022; Lee et al., 2012; Zhang et al., 2014; Ji et al., 2016; Xia et al., 2020; Liu et al., 2023; Dishi et al., 2025; Ma et al., 2018; Wang et al., 2020). Yet these obstacles are almost always confronted in isolation; the few attempts at a joint solution (Wang et al., 2020; Zhao et al., 2023) rely on architecturally rigid designs, ill-suited to continuous online learning and constraining their practical reach. This paper addresses delayed feedback and data sparsity together in a framework that is explicit, flexible, and broadly adaptable, while operating in the entire impression space to reduce selection bias.

The delayed feedback problem results from conversions that often occur long after the corresponding impression or click, leading to false negatives at training time. This is especially relevant in online settings where models update continuously on either fresh but incomplete labels, or matured but stale labels (Chapelle, 2014). The data sparsity problem refers to conversions being rare events relative to impressions or clicks, making the corresponding datasets highly imbalanced with low-signal (Ma et al., 2018). Moreover, training is typically performed only on clicked impressions, while inference utilizes the entire space of impressions. This leads to biased estimates and harms generalizability (Zadrozny, 2004).

A substantial body of work has focused on mitigating data sparsity and sample selection bias through multi-task learning and entire-space modeling. Entire Space Multi-task Model (Ma et al., 2018) addresses this by jointly learning click-through rate (CTR) and post-click CVR over the full impression space, such that the abundant click signal regularizes the sparse conversion task. This entire-space formulation has since been extended through post-click behavior decomposition and counterfactual multi-task learning (Wen

et al., 2020; Wang et al., 2022). More recent causal multi-task approaches explicitly correct for exposure and logging biases while preserving shared representations (Wang et al., 2022; Zhu et al., 2023). However, none of these approaches model the temporal structure of conversion delays, leaving them unable to exploit partially mature observations or predict when a conversion is likely to occur.

Likewise, a dedicated line of work tackles delayed feedback directly. Chapelle (2014) introduces a survival-style model to label non-converted clicks as negative or too-early-to-tell, reducing label bias under delay. Ktena et al. (2019) propose a delayed feedback loss and label-update mechanism for continuous neural network training. Yang et al. (2021) model the elapsed-time distribution of conversions using importance-sampling weights to debias training labels. However, all of these models use a single time window, which limits their ability to exploit very recent data while still accounting for long-tail conversions. Gao & Yang (2022) address this with a Multi-Head Online Learning (MHOL) framework that partitions conversion delays into mutually exclusive time windows, each handled by a dedicated head. Yet MHOL operates in a single-task, click-space setting and cannot leverage auxiliary signals or non-clicked impressions to alleviate sparsity and selection bias — leaving an opportunity to integrate MHOL with an entire-space multi-task framework.

1.1 Our Contributions

This paper proposes the flexible and model-agnostic Multi-Task Learning Multi-Head Online Learning (MTL-MHOL) framework which jointly mitigates delayed feedback and data sparsity in CVR prediction. The architecture consists of two tasks: the primary task focuses on CVR prediction through delayed feedback modeling, the auxiliary task enhances CVR prediction through the use of an engagement measure. We present the framework in the context of a multi-layer perceptron (MLP) as the base architecture and illustrate the model-agnostic behavior of our approach with a DeepFM-based implementation.

For the primary task, we adapt a multi-head model (Gao & Yang, 2022) to split the full conversion period into smaller time windows, and to train separate models for each time window using the most recent mature data available. The outputs for each model are subsequently aggregated into a single CVR estimate for more accurate predictions over the full conversion period. This way, our framework enables continuous training and updating based on the most recent observations, instead of requiring the model to wait until the complete time window has passed. Moreover, by defining the primary CVR task over the entire impression space rather than conditioning on clicks, we reduce sample selection bias caused by training on a selected subset of the inference data.

The auxiliary task is trained together with the primary task using hard parameter-sharing (Caruana, 1993). Specifically, we incorporate an engagement measure that provides extra learning signals for the model in settings where conversion data is sparse. When only a small number of observations result in a conversion, it is challenging for the model to learn from these instances. The auxiliary task therefore provides additional information during training, which helps the model learn patterns in a more robust way and makes CVR prediction more accurate, despite the sparsity of conversions.

We train and evaluate our model on data provided by a private marketing company and on a public dataset provided by the advertising company Criteo. The company datasets consist of samples of user sessions with associated conversion outcomes along with other relevant variables that may influence customer behavior collected by the company. The Criteo data contains impression, click and conversion data widely used for benchmarking CVR related models. The models are evaluated on Negative Log Loss (NLL) and Relative Cross Entropy (RCE) in a rolling window cross-validation setting.

Our MTL-MHOL model consistently outperforms both a Logistic Regression and a Random Forest model as well as all its sub-models in the ablation study with different workhorse models. The ablation study shows MTL-MHOL achieves the most consistent overall performance, and the model continues to improve as additional components are added to the framework. The model is also able to distinguish conversions that are more likely to occur soon and those likely to occur later, regardless of different temporal patterns in different datasets. Overall, the improvements demonstrate the robustness of the proposed approach to delayed feedback and the ability of our model to effectively extract information from the limited signal available.

In sum, we make the following contributions to CVR prediction in online advertising environments:

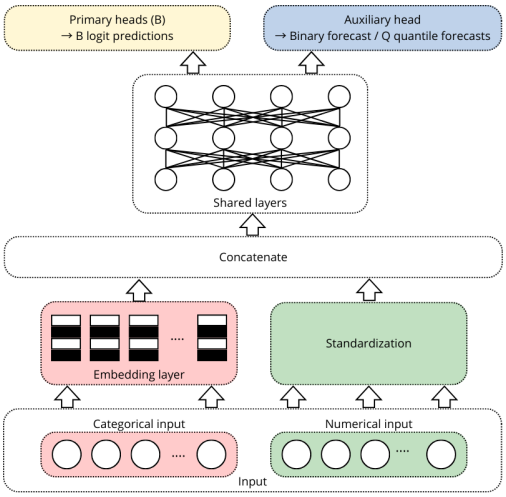


Figure 1: The proposed MTL-MHOL framework combines multi-head online learning for delayed feedback modeling with multi-task learning to incorporate auxiliary engagement signals for sparse CVR prediction. The model is illustrated using an MLP workhorse model. Categorical variables are embedded in a fixed-dimensional space. Standardized numerical values are used as one-dimensional representations and concatenated with the embeddings of categorical variables. The outputs can either be single output layers or specialized MLP towers.

- We propose the flexible and model-agnostic Multi-Task Learning Multi-Head Online Learning (MTL-MHOL) framework, which jointly mitigates delayed feedback and data sparsity in CVR prediction through the combination of multi-head online learning and multi-task learning.
- We introduce a hard parameter-sharing multi-task learning setup in which an either binary or continuous auxiliary engagement prediction task enriches delay-specific CVR prediction with additional learning signals, thereby improving robustness in sparse conversion settings.
- We show that MTL-MHOL produces temporally structured predictions across delay buckets. These capture dataset-specific conversion timing patterns and enable reliable prediction of not only whether but also when a conversion is likely to occur.
- We demonstrate through empirical analysis on proprietary company datasets and a public Criteo dataset that MTL-MHOL consistently outperforms benchmark models across NLL and RCE metrics, while remaining effective across different neural network backbone architectures.

2 Methodology

This paper proposes Multi-Task Learning Multi-Head Online Learning (MTL-MHOL) for CVR prediction. The framework combines the model-agnostic MHOL blueprint of Gao & Yang (2022) with a multi-task setting in order to jointly tackle delayed feedback and data sparsity. The MHOL design balances training data staleness and the false-negative rate in predictions, by targeting CVR prediction across mutually exclusive and collectively exhaustive delay horizons. This enables the model to incorporate information as soon as labels have matured for the shortest delay horizon, thereby ensuring that only fully matured labels are used for training. We extend this principle to a multi-task setting to enable the delay-specific heads of the MHOL component to be jointly trained with an auxiliary dense-signal task. A visualization of the model architecture is provided in Figure 1.

2.1 Workhorse Models

We present our framework in the context of a Multi-Layer Perceptron (MLP) as the base architecture. MLPs are simple feedforward networks with fully connected layers and serve as a natural and widely

adopted choice for analysis in this paper. Yet, specific modeling needs can be met by replacing the MLP with models with the desired characteristics. For example, recurrent architectures such as Long Short-Term Memory (LSTM, Hochreiter & Schmidhuber, 1997) or Gated Recurrent Unit (GRU, Cho et al., 2014) models may be employed when temporal dependencies are central, whereas models such as Deep Factorization Machines (DeepFM, Guo et al., 2017) may be preferable when capturing low-degree feature interactions is of particular importance.[1]

2.1.1 MLP

In CVR prediction, data frames commonly hold categorical and numerical features, which can be preprocessed in several different ways depending on the specific characteristics. We discuss our preprocessing pipeline, including the imputation of missing values, in Appendix B.1 and B.2. Categorical variables are represented using embedding layers that map each category to a dense vector representation, while numerical variables are used in their one-dimensional standardized representation per numerical field. These categorical and numerical representations are then concatenated to produce a single input vector per sample, which is passed to the MLP architecture.

Subsequently, the representation is processed by a shared trunk network that culminates in branches of task-specific towers. For the primary multi-head CVR prediction task, the model outputs one node per delay-specific bin, each passed through a sigmoid activation function. For the distributional auxiliary task, the model produces as many logits as quantiles to be estimated, using a linear output layer. In case of a binary auxiliary task, the corresponding tower outputs one logit that is fed into a sigmoid function.

2.1.2 DeepFM

To illustrate the adaptability of the framework, we complement our MLP implementation with a DeepFM model (Guo et al., 2017). That is the combination of the deep learning characteristics of an MLP together with the wide learning achieved by a Factorization Machine (FM, Rendle, 2010). A detailed formulation of the used implementation can be found in Appendix D. The model learns relevant low- and high-order interactions in parallel through simultaneously training the FM and Deep Neural Network components based on a joint embedding layer. For the multi-task implementation, we follow the guidelines of Yıldırım et al. (2021), who propose a biDeepFM framework for reciprocal recruitment recommender systems. We adapt this formulation slightly creating a framework that is able to allocate single as well as multiple output heads for either one of the two tasks. We note that the DeepFM framework is foremost suitable in cases of scarce and shallow interaction terms being particularly informative.

2.2 Multi-Head CVR Prediction

The delayed feedback problem arises from labels not being observed immediately after an event, leading to stale supervision when restricting training to fully matured data and increased false negatives when shorter waiting times are used. However, in online learning environments, models must be continuously updated with recent data to capture emerging trends and behavioral shifts. We therefore tackle delayed feedback with an MHOL framework.

2.2.1 Delayed Feedback Modeling

Instead of predicting a single conversion probability over the entire time horizon, we break the problem down into smaller subproblems (Gao & Yang, 2022). Specifically, the full time horizon H, during which a conversion can occur, is divided into B separate buckets such that each bucket $b \in \{1, 2, \ldots, B\}$ corresponds to a time window $w_b = (t_{b-1}, t_b]$ within $H = (t_0, t_B]$. Here, t_b denotes the end timestamp of bucket b, whereas t_0 refers to the log-timestamp of the observation. A dedicated sub-model is trained for each bucket, forming the heads of the multi-head architecture. All heads share the same input features and trunk parameters but differ in their targets, namely predicting CVR within their own time windows.

[1] Non-backpropagation backbones (e.g., kernel methods, tree ensembles) are also viable in principle (Caruana, 1997), though largely unexplored in this setting.

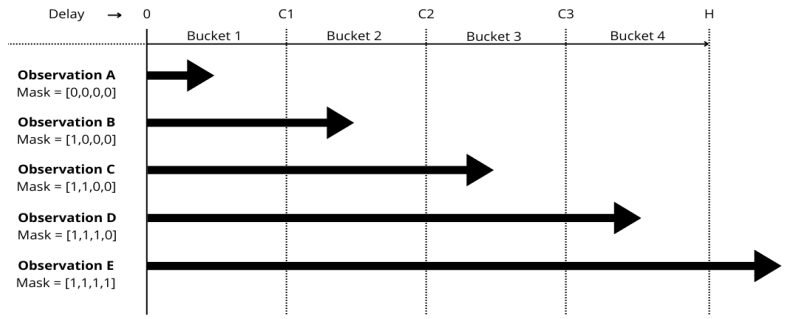


Figure 2: The five masking scenarios for a non-converter across $B = 4$ sequential delay buckets. Arrows indicate elapsed time from observation start to the current training cutoff; C_X denotes bucket cutoff $X \in \{1, 2, 3\}$; H is the maximum conversion delay.

For every observation i, we construct a vector $\mathbf{y}_i$ with B elements $[y_{i1}, \ldots, y_{iB}]$, where each element equals 1 if a conversion happened within the corresponding time window w_b, and 0 otherwise. This formulation allows the model to incorporate supervision as soon as the earliest time window w_1 has matured.

However, not all bucket labels are observed at a given cutoff time φ. We therefore employ a maturity masking approach to control which time windows contribute to training at φ. We define a maturity mask $\mathbf{m}_i = [m_{i1}, \ldots, m_{iB}]'$, where

$$m_{ib} = \begin{cases} 1, & \text{if } t_b \leq \varphi, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

indicating whether bucket b has fully elapsed by cutoff φ. For each fold, φ is set to the end of the training data. The observation's age is computed as the difference between the cutoff time and the observation's start timestamp. If no conversion occurs, a time window is marked as mature only after its full duration has elapsed. If a conversion occurs before the cutoff time, all windows are marked as mature. This ensures that negative labels are assigned only after the full time window has elapsed and prevents premature negative classification. Figure 2 illustrates the five masking scenarios that can arise for a non-converter across $B = 4$ buckets.

2.2.2 Prediction

The multi-head architecture produces B logits, one per delay bucket. For observation i, the logit for bucket b is l_{ib}; applying the sigmoid $\sigma(\cdot)$ gives the hazard probability

$$h_{ib} = \sigma(l_{ib}) = \Pr\big(y_{ib} = 1 \mid y_{i1} = \cdots = y_{i(b-1)} = 0,\, \mathcal{X}_i\big)\,, \tag{2}$$

where $y_{ib} \in \{0, 1\}$ is the conversion label for observation i in bucket b, and $\mathcal{X}_i$ is the information available at the start of observation i. The conditioning on no prior conversion is natural: by construction, at most one bucket per observation can equal one. These per-bucket hazards serve two purposes: they drive training via the loss in Section 2.4, and they provide directly interpretable online forecasts that can be updated as each bucket matures.

To aggregate across buckets, let S_i denote the survival probability, that is, the probability of no conversion through bucket B:

$$S_i = \prod_{b=1}^{B} (1 - h_{ib}). \tag{3}$$

Assuming conditional independence across buckets given $\mathcal{X}_i$, the overall predicted conversion probability is

$$\hat{p}_i = \Pr(y_i = 1 \mid \mathcal{X}_i) = 1 - S_i = 1 - \prod_{b=1}^{B} (1 - h_{ib}). \tag{4}$$

For evaluation in hindsight, $\hat{p}_i$ is compared against the realized label $y_i = \mathbb{1}\Big(\sum_{b=1}^{B} y_{ib} \neq 0\Big)$, which requires the full observation window to have elapsed.

2.3 Auxiliary Task

The multi-head architecture gains temporal resolution at a cost: splitting the conversion horizon across B buckets dilutes an already sparse signal, risking instability and overfitting within each head. We counter this dilution by introducing an auxiliary task trained jointly with the primary CVR task via hard parameter-sharing (Caruana, 1993), which lets the shared trunk learn from a denser supervision signal without altering the primary objective. A review of the benefits and risks of MTL frameworks is provided in Appendix A.1.

Depending on the dataset, the auxiliary target is either binary or continuous. For continuous targets we adopt a multi-quantile regression head for two reasons: predicting multiple quantiles yields a richer representation of the conditional distribution than a single point estimate (Ruder, 2017), and quantile regression imposes no parametric assumptions, making it robust to the skewness and heavy tails common in engagement variables.

2.4 Loss Functions

Primary task loss. Each bucket head is trained with a masked binary cross-entropy loss. Let $y_{ib} \in \{0, 1\}$ be the conversion label for observation i in bucket b, $m_{ib} \in \{0, 1\}$ the maturity mask indicating whether bucket b has elapsed by cutoff φ, and $\hat{h}_{ib} \in (0, 1)$ the predicted hazard. The loss for bucket b is

$$\mathcal{L}_{\mathrm{CE}_b} = -\frac{1}{\sum_{i=1}^{N} m_{ib}} \sum_{i=1}^{N} m_{ib} \left[y_{ib} \log(\hat{h}_{ib}) + (1 - y_{ib}) \log(1 - \hat{h}_{ib}) \right], \tag{5}$$

which is the average negative log-likelihood of a Bernoulli target over all mature observations in bucket b.

Auxiliary task loss. For a binary auxiliary target, we reuse the loss in Equation 5 with two simplifications: there is only one head $B = 1$, and the maturity mask is replaced by an auxiliary mask $a_i \in \{0, 1\}$. The mask is 1 by default and set to 0 only when a maturity timestamp is available and the auxiliary outcome is not yet observable at cutoff φ. The resulting loss is

$$\mathcal{L}_{\mathrm{aux}}^{\mathrm{binary}} = -\frac{1}{\sum_{i=1}^{N} a_i} \sum_{i=1}^{N} a_i \left[y_i^{\mathrm{aux}} \log(\hat{h}_i^{\mathrm{aux}}) + (1 - y_i^{\mathrm{aux}}) \log(1 - \hat{h}_i^{\mathrm{aux}}) \right], \tag{6}$$

where $\hat{h}_i^{\mathrm{aux}}$ is the predicted probability that $y_i^{\mathrm{aux}} = 1$.

For a continuous auxiliary target, a single point estimate would discard distributional information that could otherwise enrich the shared representation. We therefore train the auxiliary head to predict Q conditional quantiles simultaneously via a pinball loss. Let $\hat{y}_{\tau_q,i}^{\mathrm{aux}}$ be the predicted conditional quantile at level $\tau_q \in (0, 1)$, $q = 1, \ldots, Q$. The pinball loss is

$$\mathcal{L}_{\mathrm{pinball}} = \frac{1}{\sum_{i=1}^{N} a_i \cdot Q} \sum_{i=1}^{N} a_i \sum_{q=1}^{Q} \max \Big(\tau_q \, (y_i^{\mathrm{aux}} - \hat{y}_{\tau_q,i}^{\mathrm{aux}}), \, (\tau_q - 1)(y_i^{\mathrm{aux}} - \hat{y}_{\tau_q,i}^{\mathrm{aux}}) \Big), \tag{7}$$

which applies an asymmetric penalty to target each conditional quantile of the auxiliary distribution. To ensure the predicted quantiles form a valid conditional distribution, we add the soft monotonicity penalty of Decke et al. (2024) to penalize any violation of the ordering $\hat{y}_{\tau_q,i}^{\mathrm{aux}} \leq \hat{y}_{\tau_{q+1},i}^{\mathrm{aux}}$:

$$\mathcal{P} = \frac{1}{\sum_{i=1}^{N} a_i \cdot (Q - 1)} \sum_{i=1}^{N} a_i \sum_{q=1}^{Q-1} \max \Big(0, \, \hat{y}_{\tau_q,i}^{\mathrm{aux}} - \hat{y}_{\tau_{q+1},i}^{\mathrm{aux}} \Big). \tag{8}$$

The combined auxiliary loss is then

$$\mathcal{L}_{\mathrm{aux}}^{\mathrm{distr}} = \mathcal{L}_{\mathrm{pinball}} + \gamma \, \mathcal{P}, \tag{9}$$

where $\gamma \geq 0$ controls the strength of the monotonicity penalty.

Joint loss. The overall training objective for bucket b is

$$\mathcal{L}_b = \mathcal{L}_{\text{CE}_b} + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}, \tag{10}$$

where λ_{aux} scales the auxiliary contribution and $\mathcal{L}_{\text{aux}}$ is either $\mathcal{L}_{\text{aux}}^{\text{binary}}$ or $\mathcal{L}_{\text{aux}}^{\text{distr}}$. The primary loss thus remains the dominant training signal, with the auxiliary term acting as a data-adaptive regularizer. Empirically, λ_{aux} adds only marginal overhead to hyperparameter tuning. A sensitivity analysis on the Criteo data shows that for $\lambda_{\text{aux}} = [0.1, 0.5, 0.9]$, RCE varies by less than 1.2% for MTL-MHOL and less than 1.6% for MTL (see Appendix A.2 for a detailed discussion).

3 Experimental Setup

The experiments evaluate MTL-MHOL along four dimensions: benchmark comparison, component-wise ablation, bucket-level temporal accuracy, and backbone generalizability. Each dimension corresponds to one contribution and is addressed in Sections 4.1–4.4, using NLL and RCE under a rolling window cross-validation design (see Appendix C).

3.1 Data

We evaluate MTL-MHOL on two datasets: impression-level data from Criteo and session-level data provided by a private marketing company. Summary statistics for all datasets are reported in Table B.1 in Appendix B. The difference in observation unit allows us to assess the framework across complementary data regimes.

Criteo Data. We use the Attribution Modeling for Bidding dataset (Diemert et al., 2017), a widely adopted benchmark in CVR prediction research, repurposed for conversion prediction given an impression. Downsampling, preprocessing, and feature engineering are detailed in Appendix B.2.

Private Data. The private dataset covers two customers of a marketing company. Full preprocessing details are provided in Appendix B.1.

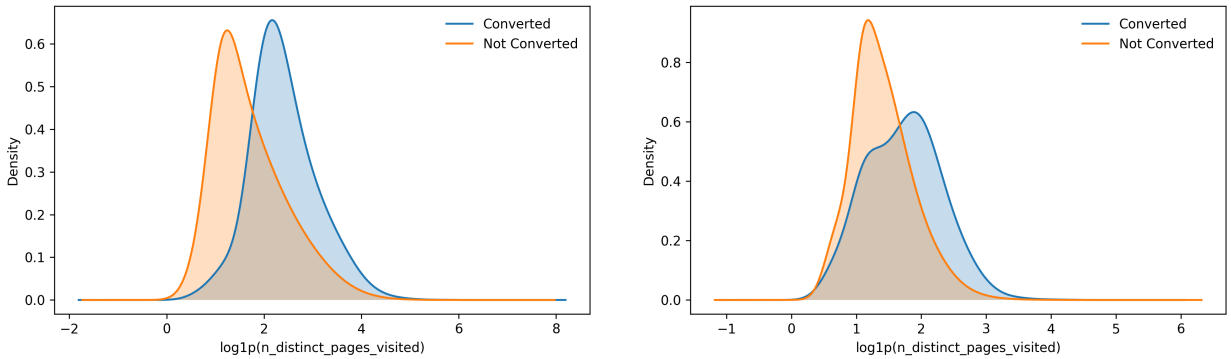
3.2 Bucket Construction

Following Section 2.2.1, the maximum delay window H is divided into four buckets, each assigned a binary conversion label. Bucket thresholds are selected from the empirical delay distribution to ensure sufficient training observations per bucket while remaining interpretable. At training time, the cutoff φ is set to the end of the training window, so that $m_{ib} = 1$ only for buckets whose upper boundary t_b satisfies $t_b \leq \varphi$. The resulting bucket boundaries and conversion distributions are reported in Appendix B.3; delay distribution plots for both datasets are in Appendix B.1 and B.2.

3.3 Auxiliary Task

Criteo Data. We use the binary click indicator as the auxiliary target. Since clicks precede conversions and are observed far more frequently, they provide a denser supervision signal that alleviates the sparsity of the primary CVR task. Organic conversions following an impression without a click cannot be modeled with the available data and are therefore excluded from this signal. The strong empirical association between clicks and conversions suggests that both tasks share underlying user behavior patterns such that representations learned from the abundant click signal transfer naturally to the rare conversion task through the shared trunk.

Private Data. Here, we use the number of distinct pages visited per session as the auxiliary target to reflect the intensity of user engagement. The variable is observed for all sessions, making it far more frequent than conversion events. Figure 3 shows that converted sessions involve more distinct page visits on average than non-converted ones. The latter are heavily concentrated at low page counts, confirming that users who browse more pages are more likely to be actively evaluating products and thus more likely to convert, suggesting shared underlying representations. The variable is nevertheless heavily right-skewed with



(a) Customer 1

(b) Customer 2

Figure 3: Kernel density of the log-transformed number of distinct pages visited by conversion status for Customers 1 and 2 after removing short sessions.

high variance, so the auxiliary target is log-transformed prior to modeling to reduce skewness and improve numerical stability. The auxiliary head predicts multiple conditional quantiles per session, with the number of quantiles treated as a tunable hyperparameter (Appendix C). Candidate quantile levels span the lower, median, and upper regions of the conditional distribution.

3.4 Architectural Choices

All MLP-based models use ReLU as the hidden-layer activation function, chosen for its computational efficiency and strong empirical performance (Guo et al., 2017). Each CVR head applies a sigmoid to its output logit to produce the bucket-specific hazard probability described in Section 2.2.2.

The MLP parameters — one weight matrix and one bias vector per layer — are learned via backpropagation. Tunable hyperparameters include the number of layers, units per layer, learning rate, dropout rate, and L2 regularization weight; full search ranges are listed in Appendix C.

The two baselines require minimal tuning. The LR has no tunable hyperparameters and the RF is fixed to the settings in Table C.2, with the exception of the minimum leaf size, which is tuned via the same nested rolling window procedure as the neural network models (Section 3.6).

3.5 Benchmark Models

We benchmark MTL-MHOL against two classical baselines: Logistic Regression (LR), a simple and widely used reference in CVR prediction, and a Random Forest (RF) as a tree-based alternative. Both produce a single CVR estimate and therefore train only on fully mature observations. All models share the same input features, with categorical variables one-hot encoded for the baselines rather than embedded.

3.6 Evaluation

We evaluate predictive performance and online learning aptitude via rolling window cross-validation. In each fold, the model is tuned on an inner rolling window covering the outer training set, retrained on the full outer training set, and evaluated against realized outcomes in the test period. For the public dataset, 5 outer folds and 2 inner folds are used; the private data are evaluated on 6 outer folds each including one inner fold due to base rate instability.

Performance is reported as mean Negative Log-Loss (NLL) and Relative Cross Entropy (RCE) across folds. Lower NLL indicates better calibrated probabilistic predictions. Higher RCE indicates a larger improvement in log-loss over a baseline that always predicts the sample mean, so a model with RCE of 37.9 reduces cross-entropy by 37.9% over this naive predictor. Full details on both metrics and the rolling window design are provided in Appendix C.

Table 1: Mean predictive performance across folds for MTL-MHOL, its ablation variants (MHOL, MTL, MLP), and the LR and RF benchmarks. Standard deviations across folds are shown in parentheses. Best result in each column in bold.

	Criteo		Customer 1		Customer 2	
	NLL (↓)	RCE (↑)	NLL (↓)	RCE (↑)	NLL (↓)	RCE (↑)
MTL-MHOL	**0.1190 (0.0082)**	**37.961 (0.894)**	**0.1318 (0.0059)**	**4.621 (1.804)**	**0.0937 (0.0060)**	**11.962 (1.467)**
MHOL	0.1219 (0.0090)	36.448 (0.971)	0.1330 (0.0059)	3.752 (1.804)	0.0949 (0.0057)	10.821 (0.666)
MTL	0.1350 (0.0052)	29.499 (2.874)	0.1331 (0.0047)	3.733 (0.525)	0.0946 (0.0058)	11.107 (1.689)
MLP	0.1362 (0.0065)	28.880 (2.098)	0.1334 (0.0079)	3.486 (1.776)	0.0953 (0.0061)	10.437 (1.926)
RF	0.1684 (0.0061)	12.049 (3.230)	0.1332 (0.0049)	3.602 (0.132)	0.0979 (0.0055)	7.981 (0.319)
LR	0.1527 (0.0068)	20.295 (2.565)	0.1353 (0.0057)	2.053 (0.865)	0.0976 (0.0058)	8.293 (1.051)

Table 2: Calibration bias across datasets. Δ denotes the absolute gap between the per-fold base rate and the average predicted probability. Lower Δ indicates better calibration.

	Criteo		Customer 1		Customer 2	
Model	mean Δ	Std. dev.	mean Δ	Std. dev.	mean Δ	Std. dev.
MTL-MHOL	0.00234	0.00211	0.00447	0.00332	0.00280	0.00325
MHOL	0.00264	0.00220	0.00505	0.00288	0.00478	0.00285
MTL	0.03230	0.00542	0.00543	0.00379	0.00547	0.00392
MLP	0.03426	0.00409	0.00648	0.00376	0.00720	0.00329
RF	0.03812	0.00729	0.00143	0.00082	0.00395	0.00086

3.7 Compute Budget

All experiments are run on a single 16-core CPU node rd-fleet.4xlarge (128GB) using PyTorch 2.7.0 on Databricks runtime 17.3LTS. Training time per outer fold is approximately 80-89 minutes for the private datasets and 145 minutes for the Criteo dataset, resulting in a total compute budget of approximately 29 CPU-hours across all datasets and folds, including hyperparameter tuning.

4 Results

4.1 Ablation Study

We conduct an ablation study to isolate the performance contribution of each architectural component. The ablation compares four variants: a shifted MLP (single head, no auxiliary task, trained only on fully mature observations), MTL (shifted MLP plus auxiliary task), MHOL (Gao & Yang, 2022) (multi-head, no auxiliary task), and the full MTL-MHOL. Comparing shifted MLP with MTL and MHOL isolates the effect of each component individually; comparing MHOL with MTL-MHOL and MTL with MTL-MHOL reveals their incremental contributions when added to the other; and comparing shifted MLP with MTL-MHOL gives the combined effect.

Component ordering. Table 1 shows that MTL-MHOL outperforms every reduced variant across all datasets. For Criteo and Customer 1, the hierarchy MLP<MTL<MHOL<MTL-MHOL emerges, with

a relative RCE gain of approximately 30% over the MLP baseline in both cases, indicating that the multi-head component accounts for the larger share of the improvement while the auxiliary task provides a consistent additional gain. For Customer 2 the ordering of the two intermediate variants reverses to MLP<MHOL<MTL<MTL-MHOL, suggesting that for this customer the auxiliary task contributes more than the additional partially mature observations. The overall RCE gain for Customer 2 is also more modest, at around 15%, approximately half that of Criteo and Customer 1.

Calibration. A key mechanism behind the multi-head advantage is calibration. Table 2 reports the average absolute deviation of predicted probabilities from the per-fold base rate. For multi-head models, this calibration error is roughly an order of magnitude smaller than for their single-head counterparts on Criteo, where the large share of partially mature observations in the training set causes single-head models to be systematically trained on a converter-enriched sample, inducing an upward bias in predicted CVR. On the private datasets (Table 2), delay horizons are short and fewer than 1% of training observations are partially mature, so this enrichment is negligible and calibration differences across variants are small, with only 0.002 and 0.004 separating the shifted MLP from MTL-MHOL for Customers 1 and 2 respectively.

Miscalibration impact. To quantify how much of the single-head deficit is attributable to miscalibration rather than lack of predictive signal, we apply an intercept-only variant of Platt scaling (Platt, 1999) to the shifted MLP on Criteo, using the unmasked final training day as a held-out calibration set. This lowers NLL to 0.121 and raises RCE to 36.694, reducing the gap to MTL-MHOL to 3.5% in RCE. MTL-MHOL achieves comparable performance without any post-hoc correction, avoiding the additional bias that arises when the base rate shifts between the calibration and test periods. The calibrated single-head figures should therefore be read as an optimistic upper bound rather than a practically realizable benchmark.

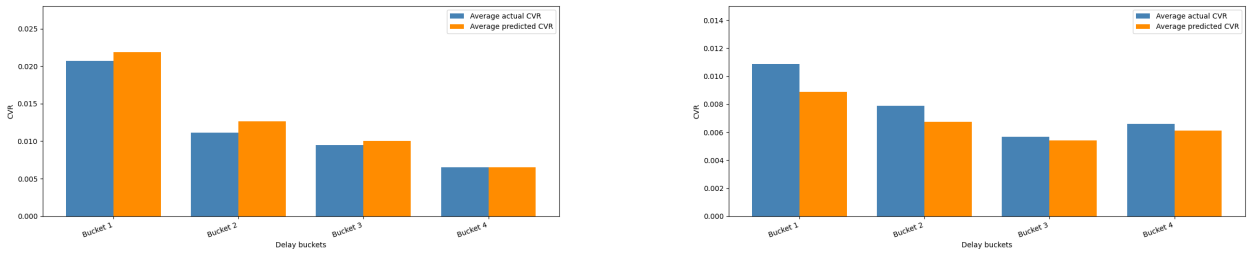
Complementarity of components. The monotone improvement pattern across all datasets and the reinforcing pattern in RCE gains support the conclusion that the two components contribute complementary information rather than overlapping signal. The multi-head architecture reduces calibration bias and introduces temporal structure through bucket-specific hazard modeling, while the auxiliary task stabilizes the shared representation via denser supervision. For Customer 2, the slightly worse calibration of MTL relative to MHOL, despite its lower NLL and higher RCE, suggests that the auxiliary task improves discrimination even when it does not reduce intercept bias.

4.2 Benchmark Comparison

Table 1 reports the predictive performance of MTL-MHOL against the LR and RF benchmarks across all datasets. MTL-MHOL outperforms both benchmarks across all datasets, with the largest absolute margins on the Criteo dataset, where the model achieves its strongest improvements in both NLL and RCE.

On the Criteo dataset, MTL-MHOL achieves 22% lower NLL and 87% higher RCE than LR, reflecting better-calibrated probabilities across both converters and non-converters. The RF is the weakest model in its raw form, with MTL-MHOL reducing NLL by 29% and roughly tripling the RCE relative to RF. Table 2 shows that this deficit is driven primarily by miscalibration: RF's average absolute deviation from the per-fold base rate is 0.038, against a mean base rate of 4.93%, with consistently positive per-fold biases indicating systematic overestimation of CVR. This overestimation stems from the stale training set, in which recent converters are included but recent non-conversions have not yet matured, inflating the positive rate that RF conditions on. Post-hoc intercept-only Platt scaling (Platt, 1999) largely corrects this bias, lifting RF to an NLL of 0.1535 and RCE of 19.956, close to LR. Allowing a slope correction as well yields an NLL of 0.1414 and RCE of 26.306, exceeding LR and approaching the single-head neural variants. MTL-MHOL, by contrast, is well calibrated without any post-hoc correction. As discussed in Section 4.1, the calibrated RF figures rely on an offline calibration set and represent an optimistic upper bound.

MTL-MHOL also outperforms both benchmarks on the private datasets. For Customer 1, it achieves 1.1% lower NLL and 28.29% higher RCE compared to RF, and achieves 2.6% lower NLL and 125% higher RCE than LR; for Customer 2, it achieves 4% lower NLL and 44% higher RCE compared to LR, and achieves 4.3% lower NLL and 49.88% lift in RCE over RF.



(a) Criteo data

(b) Private data (Customer 1)

Figure 4: Predicted CVR of MTL-MHOL compared to actual CVR for each time bucket on both datasets.

Unlike on Criteo, RF is well calibrated on the private data (Table 2). The short delay horizons mean that only a small fraction of training observations are partially mature, so the stale training set is barely enriched with converters and the upward bias in predicted CVR largely disappears.

Overall, the absolute performance gains are larger on Criteo than on the private datasets. The relative RCE improvement of MTL-MHOL over LR is highest for Customer 1, but this reflects a low RCE base rather than a larger absolute gain. MTL-MHOL is preferred across all datasets, with the most substantial benefits on Criteo.

4.3 Bucket-Level Accuracy

Figure 4 evaluates whether MTL-MHOL predicts not only if a conversion occurs but also when, by comparing predicted against actual CVR for each delay bucket, averaged across outer folds for the Criteo data (Figure 4a) and Customer 1 (Figure 4b). In both datasets, the model correctly reproduces the bucket-level hierarchy: bucket 1 consistently shows the highest CVR, and the dataset-specific temporal patterns are accurately captured. For Criteo, predicted CVR decreases monotonically across buckets, matching the observed trend. For Customer 1, CVR decreases through bucket 3 before rising again, and the model successfully tracks this non-monotone pattern as well. This indicates that MTL-MHOL is able to learn different temporal structures across datasets rather than imposing a fixed decay shape.

Residual discrepancies between predicted and actual CVR are small throughout. On Criteo, the model shows a slight upward bias across most buckets, with a maximum deviation of 0.002. On the private data, the direction reverses to a slight downward bias, with a maximum deviation of 0.003. Neither bias is large enough to affect the bucket-level ordering, and both are consistent with the aggregate calibration results in Table 2.

Overall, these results confirm that MTL-MHOL can distinguish between imminent and delayed conversions. This timing-aware behavior is directly relevant for practical applications such as budget allocation and campaign optimization, where understanding not just whether but when a conversion is likely to occur materially affects decision-making.

4.4 Model-Agnostic Evaluation

Table 3 compares the MLP and DeepFM implementations of MTL-MHOL across all datasets. On the private datasets, DeepFM outperforms the MLP variant by up to 10% in RCE; on Criteo, the MLP yields approximately 6% higher RCE.

The private data advantage of DeepFM is consistent with its feature space: 16 sparse categorical fields with many low-frequency labels — such as rarely active countries or infrequent device types — create exactly the setting where the FM component excels. By estimating latent interaction vectors, it captures informative pairwise combinations, such as a country indicator interacting with a promotional period dummy, and generalizes to feature combinations unobserved during training.

On Criteo, nine of ten categorical features are anonymized, making it difficult to assess whether meaningful pairwise interactions drive conversion. The MLP's advantage suggests that the relevant structure is better

Table 3: Mean predictive performance across folds for MTL-MHOL with DeepFM and MLP workhorse models against the LR and RF benchmarks. Standard deviations across folds are shown in parentheses. Best result in each column in bold.

	Criteo		Customer 1		Customer 2	
	NLL (↓)	RCE (↑)	NLL (↓)	RCE (↑)	NLL (↓)	RCE (↑)
MTL-MHOL (DeepFM)	0.1229 (0.0088)	35.930 (1.070)	**0.1312 (0.0060)**	**5.063 (0.542)**	**0.0932 (0.0054)**	**12.409 (0.596)**
MTL-MHOL (MLP)	**0.1190 (0.0082)**	**37.961 (0.894)**	0.1318 (0.0059)	4.621 (1.804)	0.0937 (0.0060)	11.962 (1.467)
RF	0.1684 (0.0061)	12.049 (3.230)	0.1332 (0.0049)	3.602 (0.132)	0.0979 (0.0055)	7.981 (0.319)
LR	0.1527 (0.0068)	20.295 (2.565)	0.1353 (0.0057)	2.053 (0.865)	0.0976 (0.0058)	8.293 (1.051)

captured through higher-order implicit representations than through explicit second-order interactions. Allocating model capacity to an FM component in this setting appears to constrain the higher-order layers without providing compensating signal.

Together, these results illustrate two points. First, the relative merits of each backbone are strongly data-dependent, reinforcing the recommendation in Section 2.3 to examine the feature space before committing to an architecture. Second, and more broadly, the consistent outperformance of both implementations over the benchmarks confirms that MTL-MHOL is model-agnostic within the neural network family.

5 Conclusion

We proposed MTL-MHOL, a flexible framework that jointly addresses delayed feedback and data sparsity in CVR prediction by combining a delay-aware multi-head architecture with hard parameter-sharing multi-task learning. The multi-head component partitions the conversion horizon into mutually exclusive delay buckets and estimates bucket-specific hazard probabilities via a survival-based aggregation, enabling training on partially mature observations without introducing premature negative labels. The auxiliary task enriches the sparse primary signal by jointly learning a denser engagement target through shared representations, with the auxiliary head predicting either a binary target or multiple conditional quantiles depending on the nature of the available engagement signal. The framework is evaluated on a public Criteo dataset and private data from two customers using NLL and RCE under a rolling window cross-validation design, with both MLP and DeepFM backbones and against LR and RF benchmarks.

Multi-head architecture. The multi-head component is the dominant source of performance gains. By exploiting partially mature observations through earlier heads, MTL-MHOL accesses training data entirely unavailable to single-head models, improving discrimination and yielding calibration errors an order of magnitude smaller than single-head counterparts on Criteo — without any post-hoc correction. The ablation study confirms this: the step from MLP to MHOL accounts for the largest single jump in RCE across most datasets, exceeding the contribution of any other individual component. On the private datasets, where fewer than 1% of training observations are partially mature, the gains are more modest but remain consistent across both customers, indicating that the multi-head architecture is beneficial even when the additional data it can exploit is limited.

Auxiliary task. The auxiliary task provides a consistent complementary improvement on top of the multi-head architecture across all datasets. By jointly training on a denser engagement signal, the shared trunk learns more generalizable representations, stabilizing CVR prediction in sparse settings. The ablation study shows that the step from MHOL to MTL-MHOL yields further RCE gains across all datasets, confirming that the auxiliary task contributes information beyond what the multi-head component alone can extract.

Its relative contribution is largest where the primary signal is weakest: for Customer 2, the auxiliary task accounts for a larger share of the total gain than on Criteo and Customer 1, and the ordering reverses to MLP<MHOL<MTL<MTL-MHOL, confirming that the two components are genuinely complementary rather than redundant.

Temporal structure. MTL-MHOL produces well-calibrated, temporally structured predictions across all delay buckets, correctly reproducing dataset-specific timing patterns — a monotonically decreasing CVR for Criteo and a non-monotone pattern for Customer 1. This timing-aware behavior extends the framework's utility to settings where the expected delay of a conversion affects decision-making, such as budget allocation across campaigns with different conversion cycles.

Model agnosticism. Strong performance under both MLP and DeepFM backbones confirms that MTL-MHOL is model-agnostic within the neural network family. DeepFM outperforms MLP on the private datasets, where sparse categorical fields favor explicit pairwise interaction modeling, while MLP is preferred on Criteo, where the anonymized feature space favors higher-order implicit representations. In both cases, MTL-MHOL clearly outperforms all benchmarks regardless of backbone choice.

6 Future work

Three directions are worth pursuing. First, training on larger datasets with more extensive hyperparameter searches would provide deeper insight into scalability. Second, access to click timestamps would enable more refined CTR–CVR modeling, for example by conditioning conversion prediction explicitly on clicks. Third, applying the framework to settings with longer delay horizons would more directly stress-test the multi-head architecture and is expected to yield gains beyond those observed on the private datasets.

7 Limitations

Delay horizon as the primary performance driver. The larger absolute gains on Criteo relative to the private datasets can primarily be explained by the difference in maximum delay horizon. The multi-head architecture allows MTL-MHOL to train on partially mature observations through heads that mature earlier, whereas single-head benchmarks can only use fully mature observations. This advantage scales with the delay horizon: on the private datasets, the maximum delay is about one day, thus observations mature quickly and fewer than 1% of training observations are partially mature. On Criteo, the delay window spans 12 days within a 30-day observation period, leaving roughly 50% of training observations partially mature and therefore exclusively available to MTL-MHOL. The larger pool of recent, partially mature observations directly explains the stronger gains on Criteo.

Practical implications for the private setting. The short delay horizon of the private datasets reflects a limitation of the current session-level data rather than of the framework itself. The private company employs more sophisticated session-stitching methods that would yield a longer and more realistic delay horizon. In that setting, the share of partially mature observations at training time would increase substantially, and the multi-head architecture is expected to deliver gains comparable to those observed on Criteo.

Base rate instability on private data. Another limiting factor on the private datasets is the considerable variation in base rates across folds. When the conversion rate shifts between training and test periods, the training set is a poor representation of the test set. This increases the risk of overfitting and reduces the effectiveness of hyperparameter tuning. Consistently, Customer 1 has the most stable base rates across folds and also shows the largest relative RCE improvement over LR.

Signal-to-noise ratio. The overall RCE level is lower on the private datasets than on Criteo for all models, including LR. This suggests that the private datasets contain less signal relative to noise, which compresses the absolute gains available to any model and reduces the relative advantage of a flexible architecture over a simple baseline that generalizes well under low signal conditions.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Hernán M. Alonso and Barbara Plank. When is multitask learning effective? multitask learning for semantic sequence prediction under varying data conditions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 151–161, Valletta, Malta, 2017. Association for Computational Linguistics.

Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 164–169, Valencia, Spain, 2017. Association for Computational Linguistics. doi: 10.18653/v1/E17-2026.

Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 41–48, Amherst, MA, USA, June 1993. Morgan Kaufmann. doi: 10.1016/B978-1-55860-307-3.50012-5.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

Olivier Chapelle. Modeling delayed feedback in display advertising. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1097–1105, 2014.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics, 2014.

Jens Decke, Arne Jenss, Bernhard Sick, and Christian Gruhl. An efficient multi quantile regression network with ad hoc prevention of quantile crossing. In *Proceedings of the International Conference on Architecture of Computing Systems*, pp. 51–66. Springer, 2024.

Eustache Diemert, Julien Meynet, Pierre Galland, and Damien Lefortier. Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the AdKDD and TargetAd Workshop, KDD, Halifax, NS, Canada, August, 14, 2017*, pp. 1–6. ACM, 2017.

Yuval Dishi, Ophir Friedler, Yonatan Karni, Natalia Silberstein, and Yulia Stolin. Practical multi-task learning for rare conversions in ad tech. In *Proceedings of the ACM Conference on Recommender Systems*, pp. 1042–1045, 2025.

Hui Gao and Yihan Yang. Multi-head online learning for delayed feedback modeling, 2022. arXiv:2205.12406.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1725–1731, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Wendi Ji, Xiaoling Wang, and Dell Zhang. A probabilistic multi-touch attribution model for online advertising. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pp. 1373–1382, 2016.

Shunsuke Kitada, Hitoshi Iyatomi, and Yoshifumi Seki. Conversion prediction using multi-task conditional attention networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2069–2077, 2019.

Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. Addressing delayed feedback for continuous training with neural networks in ctr prediction. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 187–195, 2019.

Kuang-Chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. Estimating conversion rate in display advertising from past performance data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 768–776, 2012.

Xiaofan Liu, Qinglin Jia, Chuhan Wu, Jingjie Li, Dai Quanyu, Lin Bo, Rui Zhang, and Ruiming Tang. Task adaptive multi-learner network for joint ctr and cvr estimation. In *Companion Proceedings of The Web Conference 2023 (WWW '23 Companion)*, pp. 490–494, Austin, TX, USA, 2023. Association for Computing Machinery. doi: 10.1145/3543873.3584653.

Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1137–1140, 2018.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alex J. Smola, Peter Bartlett, Bernhard Schölkopf, and Dale Schuurmans (eds.), *Advances in Large Margin Classifiers*, pp. 61–74. MIT Press, 1999.

Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, pp. 995–1000. IEEE, 2010. doi: 10.1109/ICDM.2010.127.

Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. arXiv:1706.05098.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2020.

Hao Wang, Tai-Wei Chang, Tianqiao Liu, Jianmin Huang, Zhichao Chen, Chao Yu, Ruopeng Li, and Wei Chu. Escm2: Entire space counterfactual multi-task model for post-click conversion rate estimation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 363–372, 2022.

Yanshi Wang, Jie Zhang, Qing Da, and Anxiang Zeng. Delayed feedback modeling for the entire space conversion rate prediction. *arXiv preprint*, 2020. arXiv:2011.11826.

Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint*, 2023. arXiv:2304.11127.

Hong Wen, Jing Zhang, Yuan Wang, Fuyu Lv, Wentian Bao, Quan Lin, and Keping Yang. Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 2377–2386, 2020.

Qianchen Xia, Jianghua Lv, Shilong Ma, Bocheng Gao, and Zhenhua Wang. A new information-theoretic method for advertisement conversion rate prediction for large-scale sparse data based on deep learning. *Entropy*, 22(6):643, 2020.

Tao Xue, Yanwu Yang, and Panyu Zhai. Conversion rate prediction in online advertising: Modeling techniques, performance evaluation and future directions. *Information Processing and Management*, 63(2): 104450, 2026.

Jia-Qi Yang, Xiang Li, Shuguang Han, Tao Zhuang, De-Chuan Zhan, Xiaoyi Zeng, and Bin Tong. Capturing delayed feedback in conversion rate prediction via elapsed-time sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4582–4589, 2021.

Ezgi Yıldırım, Payam Azad, and Şule Gündüz Öğüdücü. bideepfm: A multi-objective deep factorization machine for reciprocal recommendation. *Engineering Science and Technology, an International Journal*, 24(6):1467–1477, 2021. doi: 10.1016/j.jestch.2021.03.010.

Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 114, 2004.

Ya Zhang, Yi Wei, and Jianbiao Ren. Multi-touch attribution in online advertising with survival theory. In *Proceedings of the IEEE International Conference on Data Mining*, pp. 687–696, 2014.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022. doi: 10.1109/TKDE.2021.3070203. Original version appeared as arXiv:1707.08114.

Yunfeng Zhao, Xu Yan, Xiaoqiang Gui, Shuguang Han, Xiang-Rong Sheng, Guoxian Yu, Jufeng Chen, Zhao Xu, and Bo Zheng. Entire space cascade delayed feedback modeling for effective conversion rate prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 4981–4987, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3615475. URL https://doi.org/10.1145/3583780.3615475.

Feng Zhu, Mingjie Zhong, Xinxing Yang, Longfei Li, Lu Yu, Tiehua Zhang, Jun Zhou, Chaochao Chen, Fei Wu, Guanfeng Liu, et al. Dcmt: A direct entire-space causal multi-task framework for post-click conversion estimation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 3113–3125. IEEE, 2023.

A Methodology

A.1 Auxiliary Task Literature

Multi-task learning was first introduced by Caruana (1993), and assumes that the primary and auxiliary task share useful underlying representations (Caruana, 1993; Standley et al., 2020; Ruder, 2017). When trained jointly, this enables shared lower layers to learn generalizable features and task-specific upper layers to capture task-dependent patterns, thereby acting as a form of inductive bias that can reduce overfitting on the sparse main task (Caruana, 1997; Zhang & Yang, 2022; Ruder, 2017).

Although the literature suggests that auxiliary tasks can enhance learning of complex relationships, there is no general method for selecting tasks without risking negative transfer (Bingel & Søgaard, 2017; Ruder, 2017; Zhang & Yang, 2022). Existing hypotheses on successful task combinations remain highly context-dependent and do not generalize well across domains (Standley et al., 2020; Alonso & Plank, 2017; Bingel & Søgaard, 2017). Therefore, it is advisable to carefully base the decision on what auxiliary task to use on a series of theoretical and empirical considerations.

In the context of conversion prediction, multi-task formulations that jointly model click or other engagement signals with conversion have been shown to alleviate extreme label sparsity and improve conversion performance by letting the rare conversion task leverage denser behavioral supervision (Kitada et al., 2019; Liu et al., 2023; Dishi et al., 2025). According to Xue et al. (2026), 20 out of 34 studies report improvements over single-task models, while the remaining studies show mixed or inconsistent results. More generally, when the auxiliary variable is observed more frequently than conversion events, this allows the primary task to benefit from a richer and more informative supervision signal during training, improving representation quality for the main task (Bingel & Søgaard, 2017; Alonso & Plank, 2017).

A.2 Auxiliary Weight Sensitivity Analysis

Table A.1: Sensitivity of MTL and MTL-MHOL to the auxiliary loss weight λ_{aux} across all datasets. All hyperparameters other than λ_{aux} are held fixed. RCE is reported relative to the value at $\lambda_{\text{aux}} = 0.5$, which serves as the normalizing baseline (1.000) for each model–dataset pair.

	Criteo		Customer 1		Customer 2	
	MTL	MTL-MHOL	MTL	MTL-MHOL	MTL	MTL-MHOL
$\lambda_{\text{aux}} = 0.1$	0.967	0.969	0.836	0.958	0.939	1.074
$\lambda_{\text{aux}} = 0.5$	1.000	1.000	1.000	1.000	1.000	1.000
$\lambda_{\text{aux}} = 0.9$	1.016	1.003	1.131	0.975	1.064	0.987

Table A.1 shows that sensitivity to λ_{aux} depends on both the model variant and the degree of base rate stability across folds. For MTL-MHOL, the full-range relative RCE spread remains below 0.04 on Criteo, where base rates are stable. On the private datasets, where base rate instability is more pronounced (see Section 7), the spread widens to 0.042 and 0.074 for Customers 1 and 2 respectively, indicating that at least tuning on the first fold and fixing thereafter is preferable to using a universal a priori default. MTL is markedly more sensitive in all settings, with relative spreads of 0.164, 0.295, and 0.125 across Criteo, Customer 1, and Customer 2. In absolute terms, however, the performance gap between the best and worst λ_{aux} for MTL-MHOL corresponds to at most a 0.80-point RCE difference, which is small relative to the gains from the architectural components documented in Table 1.

A.3 Implementation Details

Throughout our analysis, we ensure a fair comparison of our proposed MTL-MHOL model and its sub-models, as well as the reported benchmark models by feeding the models with the same data preprocessing pipeline. The pipelines only diverge right before entering the model to accommodate the functional form of each model. While for neural networks we create dense embeddings, for Logistic Regression and Random Forest models we

one-hot encode the categorical features. Throughout, we use the same folds and evaluation metrics. The full code base can be found in the public github repository https://anonymous.4open.science/r/MTL-MHOL and is deployed on a Databricks environment using Databricks runtime 17.3LTS with Apache Spark 4.0.0 and Scala 2.13 running a single CPU node rd-fleet.4xlarge with 128GB of memory and 16 cores. Under these compute settings, the main model takes approximately 724 minutes to run all five outer folds, including a sequential Optuna hyperparameter optimization based on two inner folds with 32 cold start trials and 12 "fast" trials, in the case of the best set of hyperparameters of the previous outer fold resulting approximately equally strong in the current one. This amounts to approximately 145 minutes per outer fold. For the private datasets, these figures are reduced to 484 minutes and 534 minutes for Customer 1 and Customer 2, respectively. These numbers may vary substantially depending on number of folds running all 32 trials vs the quicker 12 trial reduced version, and the number of observations in the dataset. By increasing the tolerance for which the smaller number of trials is chosen, the required time can be further reduce. Furthermore, the use of a more powerful compute cluster based on a GPU could benefit overall run-time. Relevant packages and their corresponding versions are: torch (2.7.0+cpu), numpy (2.1.3), pandas (2.2.3), scikit-learn (1.6.1), optuna (3.6.1). Further details and packages can be found in the corresponding README.md file.

B Data Analysis

Table B.1: Summary statistics for all datasets. Values for the downsampled dataset used for modeling are reported first, with corresponding values for the full dataset in parentheses.

Statistic	**Criteo**	**Customer 1**	**Customer 2**
Number of observations	247,199 (16,468,027)	174,971 (8,262,943)	215,031 (4,097,427)
Number of conversions	12,181 (806,196)	5,483 (144,305)	5,030 (53,087)
Conversion rate	4.93% (4.90%)	3.13% (1.75%)	2.34% (1.30%)
Numerical predictors	4	5	5
Categorical predictors	10	16	16
Observation period	30 days	Jun 2024–Aug 2025	May 2024–Aug 2025

B.1 Private Data

The private data consist of three anonymized dataframes: the *Sessions Dataframe* (website behavior and contextual information), the *Marketing Spend Dataframe* (marketing exposure and spend), and the *Events Dataframe* (conversions and revenue). A *uid* represents a single browser cookie; a *session* is an uninterrupted website visit by a given *uid*; and an *attribution session* aggregates multiple sessions of the same *uid* not separated by long periods of inactivity. The three dataframes are merged into a single session-level dataset, where each session is the observation unit.

Downsampling. Sessions with at most one distinct page visit and a duration of five seconds or less are removed as uninformative. Additional random downsampling is then applied to obtain a computationally feasible dataset size.

Feature engineering. Three groups of features are constructed. First, within-user dynamics are captured by ordering sessions chronologically per *uid*, enabling session-index, rolling averages of historical session duration and distinct pages visited, and indicators for previous conversions. Second, temporal context variables are added: a Black Friday indicator, three-hour time-of-day categories, and day-of-week, week-of-month, and month indicators.

Preprocessing. Strictly positive numerical variables are log-transformed; variables that may contain zeros use a log-one-plus transformation. All numerical variables are then standardized using training-set parameters only. Missing numerical values are imputed to zero (the post-standardization mean) following Che et al. (2018), with a binary missingness indicator added per variable. Since the variables here lack the time-series structure of Che et al. (2018), we impute the post-standardization mean directly. The model can learn

the informational content of missingness through the associated indicator flags. Categorical variables are integer-encoded, with missing values and training-unseen categories mapped to explicit unknown indices to prevent implicit learning from categories unavailable at prediction time.

Descriptive statistics. Tables B.2 and B.3 summarize the variables used for modeling. (Note that not all variables from the original dataset are retained; marketing spend and revenue are excluded as they cannot be linked consistently.) Several categorical variables exhibit high cardinality (device and campaign identifiers) or strong concentration in a single category (country, Black Friday). Numerical variables are substantially right-skewed, reflecting large variation in user behavior.

Delay. Conversion delay is defined as the time between session start and the last recorded event timestamp. A small number of sessions with negative delays, likely due to logging errors, are truncated to zero. Figures B.1 and B.2 show the resulting delay distributions for both customers: strongly right-skewed with a small number of long delays and a maximum of approximately 18–24 hours.

Table B.2: Retained model features after downsampling and before standardizing for Customer 1.

Variable	**Number of Categories**	**Largest Share Category (%)**
Categorical Variables		
country	98	88.12
device_info_1_transformed	21	48.73
device_info_2_transformed	630	39.18
device_info_3_transformed	712	36.21
browser_info_1_transformed	24	27.21
os_info_1_transformed	8	41.39
source_transformed	291	60.86
medium_transformed	22	55.37
campaign_id_transformed	111	30.84
adset_id_transformed	418	30.84
ad_id_transformed	1330	30.84
weekday_categorical	7	17.79
month	12	12.19
week_of_month	4	33.89
time_of_day_categorical	8	20.43
black_friday	2	96.04

Variable	**Min**	**Max**	**Mean**	**Median**	**SD**
Numerical Variables					
seconds_since_last_session	0	31,325,381	377,536.869	16,648	1,298,471.051
session_number	1	3,768	4.771	1	31.869
avg_prev_pages_visited	1	280	5.66	3.167	7.529
avg_prev_session_duration	0	9,792	243.624	104.5	399.303
n_previous_conversions	0	23	0.032	0	0.219

B.2 Criteo Data

Each observation corresponds to an impression — exposure to an online advertisement — after which the individual may or may not click. A conversion can follow either a click or, less commonly, direct engagement with the ad without clicking. The observation period covers one month, significantly shorter than the private datasets.

Downsampling. Conversions are first downsampled to a maximum delay window of $H = 12$ days, since the original 30-day window would cover the full dataset and leave almost no fully mature observations available for single-head training. Non-conversions are downsampled by the same percentage to preserve the original conversion rate, after which additional random downsampling is applied for computational feasibility.

Table B.3: Retained model features after downsampling and before standardizing for Customer 2.

Variable	**Number of Categories**	**Largest Share Category (%)**
Categorical Variables		
country	147	69.33
device_info_1_transformed	27	71.11
device_info_2_transformed	911	59.87
device_info_3_transformed	1006	43.4
browser_info_1_transformed	29	36.08
os_info_1_transformed	7	60.84
source_transformed	247	52.78
medium_transformed	20	38.18
campaign_id_transformed	101	38.69
adset_id_transformed	263	38.69
ad_id_transformed	1843	38.69
weekday_categorical	7	15.68
month	12	16.04
week_of_month	4	31.12
time_of_day_categorical	8	18.69
black_friday	2	98.45

Variable	**Min**	**Max**	**Mean**	**Median**	**SD**
Numerical Variables					
seconds_since_last_session	0	34,308,807	580,157.594	47,394	1,983,867.389
session_number	1	5,981	8.609	1	141.879
avg_prev_pages_visited	1	101	3.108	2.409	2.583
avg_prev_session_duration	0	5,351	176.947	62	306.038
n_previous_conversions	0	74	0.062	0	1.448

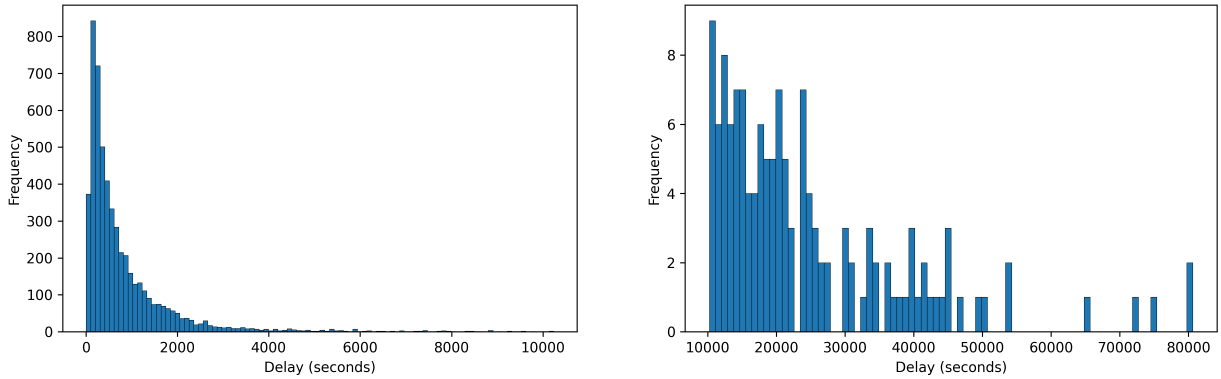


Figure B.1: Conversion Delays Customer 1, up to the 97.5th percentile (a) and the upper 2.5% tail (b), after truncation of negative delays and downsampling.

Feature engineering. Since the observation period covers only one month, time-specific variables such as Black Friday and week-of-month indicators are omitted as they are unlikely to yield additional predictive signal. The remaining features are limited to those required for training and evaluation, as reported in Table B.4.

Preprocessing. Preprocessing follows the same pipeline as for the private data (Section B.1): numerical variables are log-transformed and standardized using training-set parameters only, missing values are imputed to zero with binary missingness indicators added per variable, and categorical variables are integer-encoded with unseen categories mapped to an explicit unknown index. Timestamps are converted to datetime format to enable accurate time-based computations.

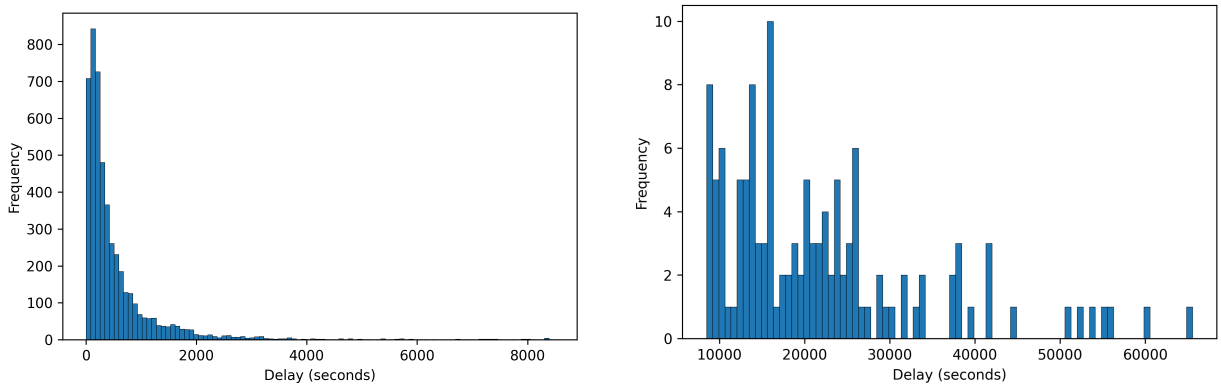


Figure B.2: Conversion Delays Customer 2, up to the 97.5th percentile (a) and the upper 2.5% tail (b), after truncation of negative delays and downsampling.

Descriptive Statistics Table B.4 summarizes the variables used for modeling. The dataset contains four numerical and ten categorical predictors, of which nine categorical features are anonymized. Categorical variables vary substantially in cardinality and concentration: cat1 and cat8 have few categories while cat3 and cat7 exhibit very high cardinality. Cat4 is heavily imbalanced with its largest category accounting for 92.77% of observations, whereas cat3 and campaign are more evenly distributed, with largest category shares of 2.01% and 2.71%. Among the numerical variables, `time_since_last_impression` and `time_since_last_click` show large ranges and high standard deviations, while `n_previous_conversions` and `impression_number` are more moderate. The conversion rate is 4.93% after downsampling, closely matching the 4.90% of the full dataset.

Delay. Figure B.3 shows the distribution of conversion delays, defined as the time in seconds from the first impression to conversion. The distribution is strongly right-skewed, with the large majority of conversions occurring within one day of initial exposure. Beyond this range, delays are sparse and spread across the full 12-day horizon, as illustrated by the upper 2.5% tail in panel (b).

Table B.4: Descriptive statistics of retained model features after downsampling and before standardizing for the Criteo data.

Variable	**Number of Categories**	**Largest Share Category (%)**
Categorical Variables		
campaign	674	2.71
cat1	9	44.43
cat2	60	52.66
cat3	1720	2.01
cat4	15	92.77
cat5	51	30.79
cat6	30	44.27
cat7	13115	13.85
cat8	11	59.63
cat9	30	53.25

Variable	**Min**	**Max**	**Mean**	**Median**	**SD**
Numerical Variables					
time_since_last_impression	0	2,619,072	287,388.097	123,908	400,459.512
time_since_last_click	0	2,591,995	270,685.446	0	526,826.543
n_previous_conversions	0	249	0.427	0	2.983
impression_number	1	865	4.479	2	8.656

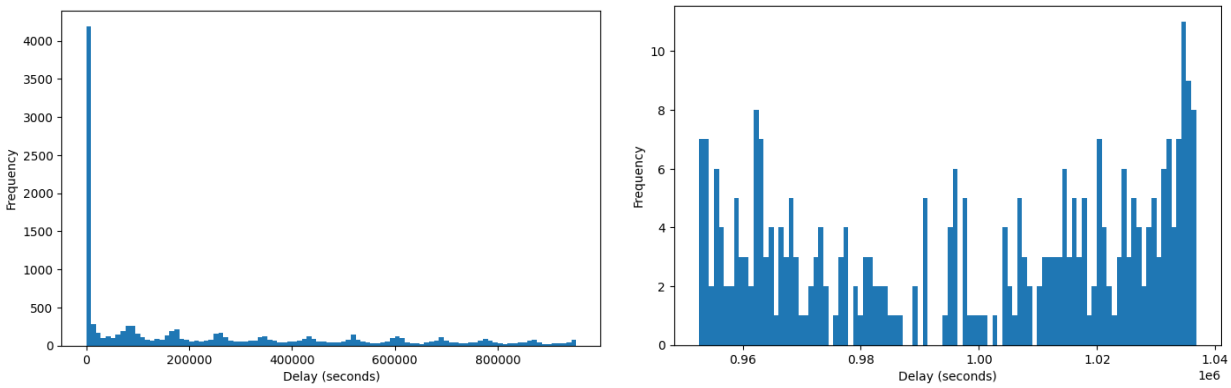


Figure B.3: Conversion Delays Criteo, up to the 97.5th percentile (a), the upper 2.5% tail (b) after downsampling.

B.3 Conversion Distribution Across Delay Buckets

Table B.5: Distribution of number of conversions across delay buckets for each customer of the private data. Percentages represent the share of total conversions within each dataset.

Bucket	Customer 1	Customer 2
0–300 sec	1809 (33.0%)	2538 (51.6%)
300–600 sec	1322 (24.1%)	1017 (20.7%)
600–1200 sec	1030 (18.8%)	725 (14.7%)
1200 sec – H	1189 (21.7%)	642 (13.0%)

Table B.6: Distribution of number of conversions across delay buckets for the Criteo dataset. Percentages represent the share of total conversions.

Bucket	Conversions
0–1 day	5466 (45.6%)
1–4 days	2773 (23.1%)
4–8 days	2247 (18.7%)
8–12 days	1507 (12.6%)

For the private data, bucket 1 corresponds to delays between 0 and 300 seconds (5 minutes), bucket 2 between 300 and 600 seconds (10 minutes), bucket 3 between 600 and 1200 seconds (20 minutes), and bucket 4 between 1200 seconds and H. This maximum conversion delay H is 80,651 seconds (22.40 hours) for Customer 1 and 65,550 seconds (18.21 hours) for Customer 2.

The Criteo dataset has a longer, more realistic maximum delay horizon of $H = 12$ days. Most conversions occur within the first day, therefore the first bucket covers delays between 0 and 1 day. The remaining buckets are chosen in a simple and intuitive way such that they still contain enough observations and roughly comparable percentages of the remaining conversions. Bucket 1 corresponds to delays between 0 and 1 day, bucket 2 between 1 and 4 days, bucket 3 between 4 and 8 days, and bucket 4 between 8 days and H.

C Evaluation

C.1 Evaluation Metrics

The models are evaluated on two probability-based evaluation metrics to capture its performance. First, we evaluate the NLL, which measures how well the predicted probabilities match the actual matured outcomes. For a binary outcome $y_i \in \{0, 1\}$ and predicted probability $\hat{p}_i$ of overall conversion, the mean NLL is given by

$$\text{NLL} \;=\; -\frac{1}{N}\sum_{i=1}^{N}\Big[\, y_i \ln(\hat{p}_i) + (1 - y_i)\ln(1 - \hat{p}_i)\Big], \tag{11}$$

where N denotes the total number of observations in the test or evaluation set. This expression corresponds to a binary cross-entropy loss $\mathcal{L}_{\text{CE}}$ (see Equation 6) for fully observed aggregated probabilities. Lower values therefore indicate better probabilistic predictions, and minimizing NLL is equivalent to minimizing the primary loss function. Here, lower values indicate better calibrated probabilistic predictions.

Furthermore, we compute the RCE, which evaluates how much better our model is compared to a simple benchmark, in our case predicting the average outcome. Let $\text{NLL}_{\text{model}}$ be the NLL of the evaluated model, and let $\text{NLL}_{\text{baseline}}$ be the NLL of a simple baseline that always predicts the sample mean $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ (with $\hat{p}_i = \bar{y}$ for all i). We define RCE as the relative improvement of the model over the baseline:

$$\text{RCE} \;=\; 1 \;-\; \frac{\text{NLL}_{\text{model}}}{\text{NLL}_{\text{baseline}}} \;=\; \frac{\text{NLL}_{\text{baseline}} - \text{NLL}_{\text{model}}}{\text{NLL}_{\text{baseline}}}. \tag{12}$$

Here, positive values indicate that the model reduces cross-entropy relative to the baseline. Moreover, both RCE and NLL penalize overconfident wrong predictions.

C.2 Rolling Window Cross Validation

The models are evaluated using a rolling window cross validation setup to ensure a realistic and time-sensitive evaluation. As opposed to random cross validation, this approach adheres to the chronological order of the data and therefore prevents data leakages from future observations into the training process.

The procedure consists of an outer and an inner loop. In the outer loop, the model is trained on a fixed window and tested on the period directly after. This window then moves forward step by step, creating multiple overlapping folds. Within each outer fold, hyperparameters are tuned using an inner cross-validation loop on the training data only. This inner loop also follows a rolling window structure, where the data is split into smaller training and validation periods while maintaining the time order to prevent leakage.

For the private datasets, in each outer fold, the model is trained on eight consecutive months and tested on the following two months. After each outer fold, the window shifts forward by one month and the procedure is repeated. This approach results in overlapping folds where the training window maintains a constant length while gradually moving forward through time. This process begins at the earliest month available in the dataset and continues until there is no longer enough data to complete a full test window.

For the Criteo dataset, online learning can be mimicked even better. Here, in each outer fold, the model is trained on the previous 25 days and tested on the following day, with a step size of one day. This reflects a setting where models are updated daily as new data becomes available.

Within each outer fold, hyperparameter tuning is performed using an inner fold cross validation procedure, where only the training observations of the corresponding outer fold are used. This training period is further divided into smaller training and validation splits for the inner folds, while again preserving the time-based order of the data. For the private datasets, we use six months for training and two months for validation for the inner folds, with a step size of one month. For the Criteo dataset, we use 23 days for training and one day for validation, with a step size of one day.

The evaluation procedure results in six outer folds for the private datasets and five outer folds for the Criteo dataset, each performing inner folds. This nested rolling window design ensures that the performance

evaluation is strictly based on past information at each step and therefore prevents data leakage from future observations. Moreover, this setup closely replicates real-world conditions, where models are trained and updated real-time to make accurate predictions.

C.3 Hyperparameter Optimization

To optimize the performance of our model, we implement hyperparameter tuning using the Tree-structured Parzen Estimator (TPE) sampler in Optuna (Watanabe, 2023; Akiba et al., 2019). By learning from past trials, it ensures an efficient tuning process and allows us to find good configurations with fewer evaluations. We complement our TPE approach by implementing the Enqueue function, which forces each fold to use the best trial from the previous fold as a prior. If in the new outer fold the old fold's optimal hyperparameter perform similarly well over all inner folds, then the number of trials is reduced. This makes the tuning process more efficient.

The parameters we tune, together with their search spaces and sampling strategies, are displayed in Table C.1. The neural-network architecture of MTL-MHOL is optimized by tuning batch size, dropout rate, learning rate, L2 weight decay, the hidden layer sizes, embedding dimensions, and the number of epochs. These choices are standard practice in optimizing the performance of a neural network model. Additionally, we tune the number of quantiles in the auxiliary task as well as the weight of the auxiliary task to the total loss function λ_{aux}. The latter not only optimizes performance, but also gives us insights into the effectiveness of complementing our multi-head model with an engagement measure to improve CVR prediction.

For each inner fold in our rolling window cross validation approach, hyperparameter tuning is performed, where by default 32 trials are executed to evaluate the optimal hyperparameters for that inner fold. In case of the enqueued hyperparameters performing strong on the new fold, too, the trials are reduced to twelve. Then, the best performing configuration of hyperparameters across all inner folds is used to train the model on the full outer fold training window before evaluating it on the outer test window. Performance is evaluated on each outer test fold, and the final reported metrics are the average across all outer folds.

We choose RCE as the objective of our hyperparameter tuning approach. By maximizing RCE, we ensure the model improves meaningfully upon a simple benchmark while providing well-calibrated predictions.

Table C.1: Hyperparameter search space used during TPE tuning with Optuna.

Hyperparameter	**Type**	**Search Space**	**Sampling**
Batch Size	Categorical	$\{256, 512, 1024, 2048\}$	Uniform
Dropout Rate	Continuous	$[0.0, 0.5]$	Uniform
Learning Rate	Continuous	$[10^{-4}, 5 \times 10^{-3}]$	Log-uniform
L2 Weight Decay	Continuous	$[10^{-7}, 10^{-3}]$	Log-uniform
Hidden Layer Sizes	Categorical	$\{(256, 256), (512, 512), (256, 128), (512, 256), (128, 128, 128), (512, 256, 128), (256, 128, 128)\}$	Uniform
Embedding Dimensions	Categorical	$\{4, 8, 16, 32\}$	Uniform
Number of Epochs	Categorical	$\{10, 11, 12, ..., 25\}$	Uniform
Number of Quantiles	Categorical	$\{3, 5, 7\}$	Uniform
Auxiliary Loss Weight	Continuous	$[0.05, 1.5]$	Log-uniform

The type of search space is determined by the nature of the hyperparameter. We employ categorical spaces in case only a predefined set of discrete options is meaningful and we use continuous spaces when the parameter can take any value within the defined range. Moreover, we apply uniform sampling when all values within the range are equally likely, whereas log-uniform sampling is used when the parameter spans several orders of magnitude and smaller values should be explored more thoroughly.

The Hyperparameters for the Random Forest benchmark model are given in Table C.2.

Table C.2: Hyperparameters (and search space) used for RF.

Hyperparameter	**Setting**
Number of Trees	500
Maximum depth	None
Minimum Samples per Split	2
Minimum Samples per Leaf	$\{50, 100, 150, 200, 250\}$
Maximum Features Considered per Split	square root of total features
Maximum Proportion of Samples in each Tree	0.7

D Appendix D - DeepFM

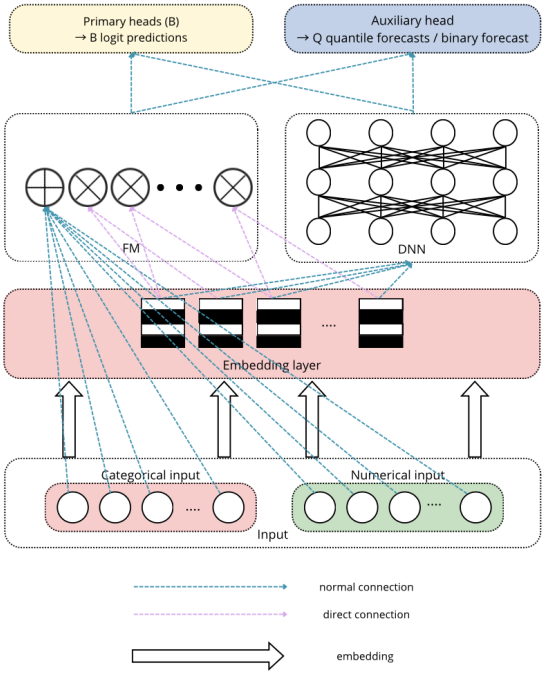


Figure D.1: The DeepFM architecture visualized. A blue arrow represents a normal connection and is defined as a connection with weights to be learned by the model. These weights can include matrices for projections onto the desired dimensional space. Direct connections in pink are connections with weight 1 by default. Thick white arrows indicate embeddings based on the preprocessed inputs. A + in a circle represents a sum and × indicates a dot-product. The empty circles in the MLP can be any nonlinear activation function chosen by the researcher. Here, we choose ReLU activation functions.

Following Guo et al. (2017), we use a shared embedding architecture that combines an FM component for first- and second-order effects with a feed-forward network for higher-order effects. In the original DeepFM formulation for CTR prediction, both parts share the same raw input and the same latent feature embeddings, and the final click probability is obtained from the sum of the FM and Deep Neural Network logits (Guo et al., 2017). In the multi-task extension of Yıldırım et al. (2021), the same shared network is equipped with multiple output heads and optimized by a weighted sum of task losses, where the task-specific parameters are confined to the final layer.

Let an observation $(\mathbf{x}_i, y_i)$ consist of F_c categorical fields and F_n numerical input dimensions. I.e., we can write the covariates of observation i as $\mathbf{x}_i = [x_{i,1}, \ldots, x_{i,F}]$, with $F = F_c + F_n$. Let $j \in \{1, \ldots, F\}$ denote the specific field considered. For categorical field $j \in \{1, \ldots, F_c\}$, let $x_{i,j}^{(c)} \in \{0, \ldots, S_j - 1\}$ denote its encoded level, where index 0 is reserved for unseen values at evaluation time. The model learns a first-order scalar weight look-up table $\mathbf{W}_j^{(c)} \in \mathbb{R}^{S_j \times 1}$, such that $w_{i,j}^{(c)} = \mathbf{W}_j^{(c)}[x_{i,j}^{(c)}] \in \mathbb{R}$. It also learns a second order look-up table $\mathbf{V}_j^{(c)} \in \mathbb{R}^{S_j \times d}$, yielding a d-dimensional latent embedding $\mathbf{e}_{i,j}^{(c)} = \mathbf{V}_j^{(c)}[x_{i,j}^{(c)}] \in \mathbb{R}^d$. For numerical input $x_{i,j}^{(n)} \in \mathbb{R}$, raw numeric covariates are preprocessed by a train-set transformation (log or $\log(1+x)$), standardized, and augmented with a missingness indicator, as described in Section B.1. Further, each resulting scalar input is embedded multiplicatively as

$$\mathbf{e}_{i,j}^{(n)} = x_{i,j}^{(n)} \mathbf{v}_j^{(n)}, \qquad \mathbf{v}_j^{(n)} \in \mathbb{R}^d,$$

rather than being discretized or passed as a one-hot field as in the generic descriptions of Guo et al. (2017) and Yıldırım et al. (2021). This is done for efficiency purposes, such that in the FM component latent vectors do not need to be rescaled by the corresponding raw numeric values, yet represents no theoretic alteration of the model. Additionally, a first-order scalar weight $w_j^{(n)} \in \mathbb{R}$ is learned.

Collecting all field embeddings as $\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,F} \in \mathbb{R}^d$, the FM component is

$$s_{\mathrm{FM}}(\mathbf{x}_i) = b + \sum_{j=1}^{F_c} w_{i,j}^{(c)} + \sum_{j=1}^{F_n} w_j^{(n)} x_{i,j}^{(n)} + \sum_{r=1}^{F} \sum_{s=r+1}^{F} \langle \mathbf{e}_{i,r}, \mathbf{e}_{i,s} \rangle, \tag{13}$$

where $\langle \mathbf{e}_{i,r}, \mathbf{e}_{i,s} \rangle$ denotes the dot product between the two latent vectors, and b is the learned bias.

This formulation matches the standard DeepFM/FM decomposition into linear and pairwise interaction terms. The deep component uses the same embeddings, flattened and passed through a multi-layer perceptron with R output heads,

$$\eta(\mathbf{x}_i) = \mathrm{MLP}(\mathrm{vec}(\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,F})) \in \mathbb{R}^R,$$

following the shared-embedding principle of DeepFM.

The present implementation differs slightly from the canonical presentation $\sigma(y_{\mathrm{FM}} + y_{\mathrm{DNN}})$ in Guo et al. (2017): instead of producing one FM logit and one Deep Neural Network logit and summing them directly, it first forms a shared scalar $s_{\mathrm{FM}}(\mathbf{x}_i)$ and a shared hidden representation $\eta(\mathbf{x}_i)$, and then applies task-specific linear maps. For the primary task with B heads,

$$\mathbf{z}^{\mathrm{prim}}(\mathbf{x}_i) = \mathbf{A}_{\mathrm{FM}}^{\mathrm{prim}} s_{\mathrm{FM}}(\mathbf{x}_i) + \mathbf{A}_{\mathrm{DNN}}^{\mathrm{prim}} \eta(\mathbf{x}_i) + \mathbf{b}^{\mathrm{prim}} \in \mathbb{R}^B.$$

When $B = 1$, this reduces to an ordinary single-logit binary model. When $B > 1$, each component $z_b^{\mathrm{prim}}(\mathbf{x}_i)$ is interpreted as a bucket-specific logit and the observation-level conversion probability is aggregated at evaluation time as

$$\hat{p}(\mathbf{x}_i) = 1 - \prod_{k=1}^{K} \left(1 - \sigma\left(z_b^{\mathrm{prim}}(\mathbf{x}_i)\right)\right).$$

The per bucket training loss for the primary task is given in Equation 5.

Auxiliary tasks are added in the spirit of the multi-objective formulation of Yıldırım et al. (2021), but with a broader set of heads than considered there. For a binary auxiliary target, the model outputs

$$z^{\mathrm{aux}}(\mathbf{x}_i) = a_{\mathrm{FM}} s_{\mathrm{FM}}(\mathbf{x}_i) + \mathbf{a}_{\mathrm{DNN}}^{\top} \eta(\mathbf{x}_i) + b_{\mathrm{aux}},$$

and optimizes a masked binary cross-entropy loss (see Equation 6). For a quantile auxiliary target with quantile levels $\tau_1 < \cdots < \tau_Q$, the model outputs $\mathbf{q}(\mathbf{x}_i) \in \mathbb{R}^Q$ with

$$\mathbf{q}(\mathbf{x}_i) = \mathbf{A}_{\mathrm{FM}}^{\mathrm{aux}} s_{\mathrm{FM}}(\mathbf{x}_i) + \mathbf{A}_{\mathrm{DNN}}^{\mathrm{aux}} \eta(\mathbf{x}_i) + \mathbf{b}^{\mathrm{aux}},$$

and minimizes a masked pinball loss (see Equation 7) together with a monotonicity penalty enforcing $q_1(\mathbf{x}_i) \leq \cdots \leq q_Q(\mathbf{x}_i)$ (see Equation 8). Hence the total objective mirrors the implementation of the MLP version (see Section 2.4). Thus, relative to Guo et al. (2017), our implementation preserves the essential shared-embedding DeepFM structure but generalizes the final prediction layer to task-specific heads. Relative to Yıldırım et al. (2021), it retains the multi-objective hard parameter-sharing idea while replacing the original dual-sigmoid setup by a more general family of single-head, multi-head, binary-auxiliary, and quantile-auxiliary objectives.