FRI Data Science Project Competition 2024
Final report

State-of-the-art CTR and CVR Architectures

Rok Rajher, Jan Kuhta, Gašper Spagnolo

Abstract

Our collaborative research with Outbrain aimed to enhance Click-Through Rate (CTR) and Conversion Rate (CVR) prediction in online advertising. We evaluated various architectures, including Factorization Machines, Logistic Regression, Deep Neural Networks, DeepFM, and Deep & Cross Networks, as well as implemented state-of-the-art models such as Attention Factorization Machines, Gated Deep & Cross Network, Enhanced Deep & Cross Network, and DCN-Mix. Furthermore, we developed a batch skipping approach with which we were able to improve models' performance on both datasets. Among the models tested, DeepFM consistently outperformed others, achieving AUC scores of 0.8379 (CTR) and 0.9175 (CVR) without batch skipping, and 0.8380 (CTR) and 0.9179 (CVR) with batch skipping.

Keywords
CTR prediction, CVR prediction, Online learning

Advisors: izr. prof. dr. Tomaž Curk, Samo Pahor, Blaž Mramor

Introduction

In our collaborative research with the company Outbrain, we have explored prediction models for Click-Through Rate (CTR) and Conversion Rate (CVR) in online advertising. CTR is a key metric in online advertising, measuring the proportion of users who click on an ad compared to those who view it. CVR measures the percentage of users who take a desired action out of the total number of users who click on an ad. Both are crucial indicators of ad effectiveness, reflecting audience engagement and ad relevance.

For Outbrain, optimizing CTR and CVR prediction models is crucial in guiding their advertising strategy. Accurate predictions enable the strategic allocation of resources, the customization of ad content, and the optimization of bidding strategies to enhance ad placement and maximize exposure on websites.

Our initial research focused on establishing a reliable testing environment, exploring state-of-the-art CTR prediction models, and fine-tuning their hyperparameters. This process provided valuable insights into current practices and helped us establish a baseline model. Building on this foundation, we incorporated CVR predictors and shifted our focus to developing new, modern models for both CTR and CVR predictions. Additionally, we optimized the training and evaluation processes to better suit the online learning environment, ensuring more accurate and efficient model performance.

Methods

CTR Dataset

The company supplied us with a segment of anonymized real data, consisting of 11.442.641 samples with 20 features. This dataset is structured in a time sequence, spanning over three weeks. Each value has been pre-processed using a company-specific hash function. Specifics of each feature are undisclosed, except for the target variable indicating whether the ad was clicked or not. The dataset has also been rebalanced, with approximately 35% of the data representing positive instances (ad clicks) and 65% representing negative instances, whereas in real-world scenarios, positive instances of ad clicks typically occur at around 1%.

It is important to note that the data is sourced exclusively from ads displayed in applications.

CVR Dataset

Similarly to CTR dataset, the data is anonymized, but slightly larger, consisting of 22.680.015 samples with 35 features. This dataset is structured in a time sequence, spanning over 6 week period. Each value has been pre-processed using a company-specific hash function. Specifics of each feature are undisclosed, except for the target variable indicating whether the ad was converted or not.

However, this dataset has not been rebalanced, with approximately 10% of the data representing positive instances (ad convertions) and 90% representing negative instances.

The data is sourced from both applications and websites.

Models

Our research incorporated two standard machine learning methods: Logistic Regression (LR) and Neural Networks (NN). Additionally, Outbrain provided us with some popular models currently used in the industry for CTR and CVR prediction: Factorization Machines (FM) [1], DeepFM [2], and Deep & Cross Network (DCN) [3].

Building upon these models, we developed and improved new state-of-the-art models, including Attention Factorization Machines (AFM) and Gated Deep & Cross Network (GDCN), Enhanced Deep & Cross Network (EDCN) and DNC-Mix.

DNN

Our DNN model has a standard deep neural network architecture. It initially flattens the input to a single dimension vector. It then sequentially adds a specified number of hidden layers. Each hidden layer uses the ReLU activation function and the Glorot normal initializer for weight initialization. To mitigate overfitting, dropout is applied to each dense layer. The network concludes with an output layer that produces a single logit value.

Attention Factorization Machines (AFM)

AFM enhances traditional Factorization Machines by integrating an attention mechanism to focus on relevant feature interactions, improving prediction accuracy. The attention mechanism assigns weights to feature interactions based on their importance.

Formally, the attention network is defined as:

$$a_{ij} = \frac{\exp(h^T \mathrm{ReLU}(W(v_i \odot v_j)x_i x_j + b))}{\sum_{(i,j)\in R_x} \exp(h^T \mathrm{ReLU}(W(v_i \odot v_j)x_i x_j + b))},$$

where W, b, and h are model parameters, t denotes the hidden layer size of the attention network, v_i and v_j are feature vectors, and x_i and x_j are feature values.

The attention scores a_{ij} are normalized using the softmax function. The output of the attention-based pooling layer is a k-dimensional vector, compressing all feature interactions in the embedding space by distinguishing their importance.

The overall formulation of the AFM model is:

$$\hat{y}_{\mathrm{AFM}}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + p^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}(v_i \odot v_j)x_i x_j,$$

where p is a projection vector and $\{w_i\}_{i=0}^n$ represents the set of weights of the linear layer, both being parameters of the model [4].

DCN

Deep & Cross Network [3] is a neural network architecture that combines a deep component and a cross component to capture both low-order and high-order feature interactions. The deep component resembles a traditional feedforward neural network, while the cross component explicitly captures feature interactions through cross-layer connections.

In our research, we implemented a stacked version of DCN, where the output of the cross layer is used as an input of the deep layer. The final output of the model is calculated using the following formula:

$$\hat{y} = \sigma(\phi_{\mathrm{L2}}(\mathbf{x}_{\mathrm{L1}}; \theta)),$$

where $\hat{y}$ represents the predicted output, $\mathbf{x}_{\mathrm{L1}}$ detones outputs from the cross layer, ϕ_{L2} the output of the neural network layer parameterized by θ, with $\sigma(x) = \frac{1}{1+\exp(-x)}$.

Gated Deep & Cross Network (GDCN)

Gated Deep & Cross Network (GDCN) integrates the strengths of Deep & Cross Network (DCN) with a gating mechanism, effectively capturing both low-order and high-order feature interactions. This architecture employs a core structure called the Gated Cross Network (GCN), which models explicit bounded-degree feature crosses with an information gate.

The $(l+1)^{\mathrm{th}}$ gated cross layer of the GCN is represented as follows:

$$c_{l+1} = c_0 \odot \left(W_l^{(c)} \times c_l + b_l\right) \odot \sigma\left(W_l^{(g)} \times c_l\right) + c_l,$$

Where the variable c_0 is the base input from the embedding layer, containing 1st-order features. c_l represents the output features from the previous l^{th} gated cross layer and serves as input to the current $(l+1)^{\mathrm{th}}$ layer in $\mathbb{R}^D$. The matrices $W_l^{(c)}$ and $W_l^{(g)}$ are $D \times D$ learnable parameters, b_l is the bias vector in $\mathbb{R}^D$, $\sigma(\cdot)$ denotes the sigmoid function, and $\odot$ signifies the element-wise product.

GDCN has two main structural variants: stacked (GDCN-S) and parallel (GDCN-P). In GDCN-S, the embedding vector c_0 is fed into the GCN, and its output is subsequently processed by a DNN. Conversely, in GDCN-P, c_0 is simultaneously inputted into both the GCN and DNN, and their outputs are concatenated. We implemented both versions of the model.

Enhanced Deep & Cross Network (EDCN)

EDCN improves upon the standard Deep & Cross Network (DCN) through two key innovations: the Bridge Module and the Regulation Module.

Bridge Module: Unlike DCN's late fusion approach, EDCN employs a dense fusion strategy that shares information at each layer, enhancing interaction between explicit and implicit feature modeling. This module captures layer-wise interactive signals between the cross and deep layers using various interaction functions (e.g., Pointwise Addition, Hadamard Product, Concatenation, Attention Pooling), thus mitigating gradient skew issues during backpropagation.

Regulation Module: EDCN introduces a field-wise gating mechanism that selectively regulates features fed into the parallel networks. Instead of equally distributing features, this module uses a Softmax-based gating system to assign the most relevant features to each network, improving the modeling of feature interactions.

These enhancements enable EDCN to capture more nuanced interactions and improve the overall learning process compared to the traditional DCN.

DCN-Mix

The DCN-Mix introduces significant enhancements over the standard DCN by incorporating a mixture of experts (MoE) [5] approach and leveraging low-dimensional projections for improved feature interaction modeling. These experts are combined using a gating mechanism that adapts based on the input.

While standard DCN directly projects the input back to the original high-dimensional space after learning feature crosses, DCN-Mix applies additional nonlinear transformations within the low-dimensional projected space to refine the representation before projecting back to the original space. The formulation is:

These improvements make DCN-Mix more capable of capturing complex feature interactions and enhancing model performance, particularly in the context of constrained memory and time budgets [6].

Metric selection

In our collaboration with Outbrain, we prioritized the predictive probabilities generated by the CTR models over binary classifications. This emphasis on probabilities aligns with Outbrain's strategy, as it enables tailored advertising approaches without the need for setting rigid thresholds.

Given this focus, our evaluation of the CTR models centered on metrics that leverage the probabilistic nature of the predictions: AUC, and LogLoss.

Models' hyperparameter selection

To standardize hyperparameters across models, the parameter ranges were uniformly set for all models. Learning rates varied from 0.001 to 0.02 and L2 regularization from 0 to 0.1. For deep neural networks architectures, we used dropout rates from 0 to 0.5, 1 to 5 hidden layers, each with 50 to 200 neurons, and ReLU activation. For FM architectures, we varied the latent space size k from 3 to 10. Specifically for GDCN, we varied a gated layer size ranging from 2 to 6. For DCN-Mix number of experts ranged from 4 to 16. For EDCN, GDCN, and DCN-Mix, we regularized embeddings with a factor τ ranging from 0.05 to 1.2. All models utilized the Lazy Adam optimizer.

Training and evaluation process

Our initial training and evaluation process followed the traditional train-validation-test approach. However, to better align with the online learning environment, we adopted a different evaluation technique. We simulated Outbrain's real-world online batch learning scenario by initially predicting with batch data and then training our model in small batches of 1,000 instances each, representing approximately a 3-minute span of data. This method allowed us to track metrics such as AUC and LogLoss after every batch, enabling us to measure the success of our models based on the averages of these metrics throughout the batch learning process.

After hyperparameter tuning on the training set using the Python library *Optuna*, we evaluated the performance on the test set similarly, but to more accurately capture the models' uncertainty, we used bootstrapping. This involved continuous training on the bootstrapped test set, simulating real-world conditions. Consequently, our final results are averages across all the batches in the test set, providing a robust evaluation of model performance in an online learning context.

Batch skipping

During the evaluation process, we observed an interesting trend in our AUC and LogLoss scores for CTR predictions, as shown in Figure 1. While GDCN-S is highlighted as an example, this trend was consistent across all models. Our scores appeared to periodically drop after approximately 500 batches, which corresponds to roughly a 24-hour period. To address this issue, we experimented with various techniques. The most effective solution was to skip training the models on n consecutive batches after the current batch AUC surpassed the *auc_threshold*, with n and *auc_threshold* being hyperparameters that we also tuned. This adjustment helped improve the overall performance of the majority of our models.

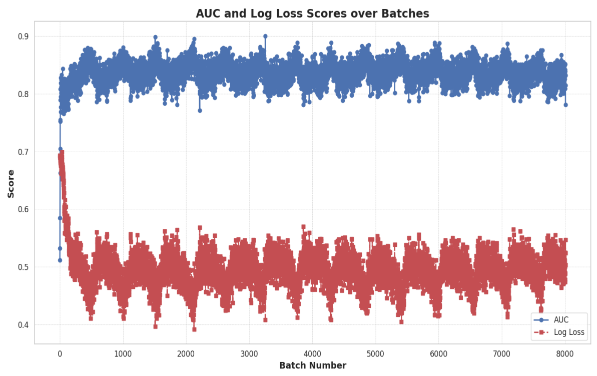

Figure 1. AUC and LogLoss Scores During GDCN-S Training on the CTR Dataset

Results

Performance metrics of our models are displayed in table 1 for CTR data and table 2 for CVR data. We reported AUC and LogLoss scores, along with their standard deviations, obtained from 100 bootstrap samples.The first two columns in each table show the results for our base models, while the second two columns show the results for models with batch skipping approach.

Table 1. Performance of Models on CTR dataset

	Base		With skipping	
Model	**AUC**	**LogLoss**	**AUC**	**LogLoss**
DeepFM	**0.8379 ± 0.0004**	**0.4909 ± 0.0005**	**0.8380 ± 0.0004**	**0.4908 ± 0.0005**
DNN	0.8378 ± 0.0003	0.4912 ± 0.0006	0.8380 ± 0.0003	0.4911 ± 0.0005
GDCN-P	0.8371 ± 0.0003	0.4920 ± 0.0005	0.8370 ± 0.0003	0.4921 ± 0.0005
D&C	0.8371 ± 0.0004	0.4921 ± 0.0005	0.8372 ± 0.0004	0.4919 ± 0.0006
GDCN-S	0.8371 ± 0.0004	0.4922 ± 0.0005	0.8370 ± 0.0003	0.4920 ± 0.0005
DCN-mix	0.8371 ± 0.0004	0.4922 ± 0.0005	0.8369 ± 0.0003	0.4922 ± 0.0005
LR	0.8365 ± 0.0003	0.4926 ± 0.0006	0.8365 ± 0.0004	0.4923 ± 0.0006
AFM	0.8365 ± 0.0004	0.4928 ± 0.0005	0.8367 ± 0.0004	0.4923 ± 0.0005
FM	0.8361 ± 0.0004	0.4931 ± 0.0005	0.8363 ± 0.0003	0.4925 ± 0.0006
EDCN	0.8313 ± 0.0005	0.5126 ± 0.0006	0.8332 ± 0.0004	0.6427 ± 0.0007

CTR

For the CTR dataset, the DeepFM model achieved the highest AUC (0.8379) and lowest LogLoss (0.4909) among the base models, closely followed by the Deep Neural Network, with Deep and Cross models trailing. With batch skipping, DeepFM showed marginal improvements to an AUC of 0.8380 and LogLoss of 0.4908. Other models demonstrated similar improvements or remained consistent, indicating that our batch skipping approach is effective on the CTR dataset.

Table 2. Performance of Models on CVR dataset

	Base		With skipping	
Model	**AUC**	**LogLoss**	**AUC**	**LogLoss**
DeepFM	**0.9175 ± 0.0003**	**0.2234 ± 0.0004**	**0.9179 ± 0.0003**	**0.2228 ± 0.0004**
DNN	0.9174 ± 0.0002	0.2237 ± 0.0003	0.9178 ± 0.0003	0.2232 ± 0.0005
D&C	0.9168 ± 0.0003	0.2249 ± 0.0003	0.9173 ± 0.0003	0.2236 ± 0.0005
GDCN-P	0.9164 ± 0.0002	0.2250 ± 0.0004	0.9168 ± 0.0003	0.2242 ± 0.0004
GDCN-S	0.9164 ± 0.0003	0.2250 ± 0.0004	0.9170 ± 0.0004	0.2239 ± 0.0004
DCN-mix	0.9163 ± 0.0004	0.2251 ± 0.0004	0.9168 ± 0.0002	0.2242 ± 0.0004
LR	0.9149 ± 0.0003	0.2264 ± 0.0003	0.9148 ± 0.0003	0.2264 ± 0.0005
AFM	0.9141 ± 0.0004	0.2282 ± 0.0005	0.9162 ± 0.0003	0.2251 ± 0.0005
FM	0.9138 ± 0.0003	0.2300 ± 0.0004	0.9147 ± 0.0003	0.2289 ± 0.0004
EDCN	0.9006 ± 0.0003	0.2911 ± 0.0004	0.9107 ± 0.0003	0.2689 ± 0.0005

CVR

For the CVR dataset, the DeepFM model demonstrated the best performance with an AUC of 0.9175 and a LogLoss of 0.2234, closely followed by the Deep Neural Network and then Deep and Cross models in the base configuration. With batch skipping, DeepFM further improved to an AUC of 0.9179 and a LogLoss of 0.2228. Other models showed similar improvements, indicating that the batch skipping approach is effective for CVR datasets as well.

Interpretation

While model performance differences appear minimal in offline AUC scores, even small improvements can significantly enhance online metrics like Online Acquisition Gain. This occurs because online environments allow models to interact with live user data, capturing real-time behaviors and preferences. Consequently, a slightly better model can produce more relevant recommendations or advertisements, leading to higher acquisition rates. This effect is especially pronounced in online systems where continuous learning and user feedback enable dynamic performance optimization. Thus, offline AUC scores provide a baseline, but the true impact of model improvements is more evident in live, interactive settings [7].

Discussion

In this report, we evaluated the performance of various models on both CTR and CVR datasets, comparing their AUC and LogLoss metrics in base configurations and with the application of batch skipping. Our findings indicate that the DeepFM model consistently outperformed other models across both datasets, achieving the highest AUC and lowest LogLoss.

For the CTR dataset, DeepFM reached an AUC of 0.8379 and a LogLoss of 0.4909, with marginal improvements to 0.8380 and 0.4908 when batch skipping was applied. Similar trends were observed in the CVR dataset, where DeepFM achieved an AUC of 0.9175 and a LogLoss of 0.2234, further improving to 0.9179 and 0.2228 with batch skipping. Other models also demonstrated slight enhancements with batch skipping, confirming its effectiveness.

In conclusion, although none of our model implementations outperformed the models provided by Outbrain, our implementation of batch skipping proved beneficial. It enhanced model performance across both CTR and CVR datasets, making it a promising area for future exploration and development.

References

[1] Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

[2] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[3] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7. 2017.

[4] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks, 2017.

[5] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[6] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep &amp; cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, WWW '21. ACM, April 2021.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.