Ad-Rec: Advanced Feature Interactions to Address Covariate-Shifts in Recommendation Networks

Muhammad Adnan[⋆] **Yassaman E. Maboud**[⋆] **Divya Mahajan**[†] **Prashant J. Nair**[⋆]
The University of British Columbia[⋆] Georgia Institute of Technology[†]
{adnan,yassaman,prashantnair}@ece.ubc.ca divya.mahajan@gatech.edu

Abstract

Recommendation models enhance user experiences by utilizing input feature correlations. However, deep learning-based models encounter challenges from changing user behavior and item features, leading to data distribution shifts. Effective cross-feature learning is crucial in addressing this. We introduce Ad-Rec, an advanced network that leverages feature interaction techniques to tackle these issues. It utilizes masked transformers to learn higher-order cross-features while mitigating data distribution drift. Our approach improves model quality, accelerates convergence, and reduces training time. We demonstrate scalability of Ad-Rec and its superior model quality through extensive ablation studies.

1 Introduction

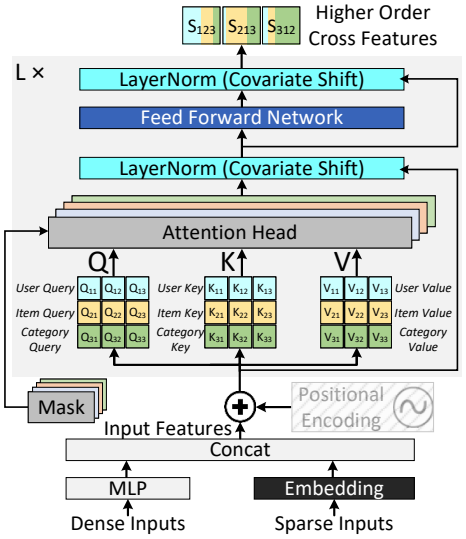
Recommendation models, essential for personalized recommendations in web services, have evolved from traditional collaborative filtering to deep learning-based approaches like neural collaborative filtering [Koren et al., 2009, He et al., 2017, Naumov et al., 2019, Zhao et al., 2019, Ishkhanov et al., 2020]. However, deep learning models face challenges due to dynamic user behavior and evolving item features, causing distribution shifts and performance degradation [Naumov et al., 2019, Ishkhanov et al., 2020, Guo et al., 2017]. This paper aims to address these challenges.

Deep learning recommendation models include neural networks, embedding lookups, and feature interactions. Feature interactions integrate latent representations from various inputs to generate personalized recommendations. While non-sequential models rely on user activity, sequential and session-based models consider historical interactions, to account for order and context.

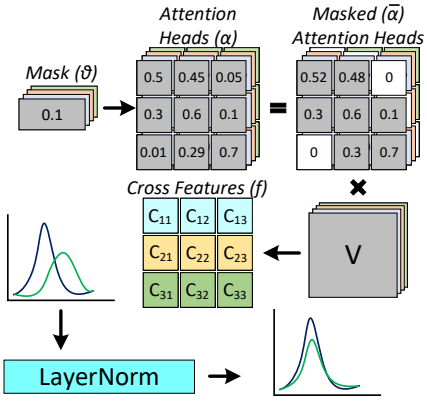
Feature interaction is vital in recommendation tasks, enhancing both non-sequential and sequential models with valuable insights [Naumov et al., 2019, Ishkhanov et al., 2020, Guo et al., 2017]. However, deep learning-based recommendation models face challenges due to covariate shifts. Manually identifying cross-features becomes impractical with numerous features, hindering model generalization. To address this, deep neural network (DNN) based feature interaction techniques have emerged [Cheng et al., 2016, Guo et al., 2017, Lian et al., 2018, Shan et al., 2016, Song et al., 2019, Chen et al., 2019, Li et al., 2019], enabling the extraction of higher-order features. Yet, including irrelevant feature interactions can introduce noise and overfitting [Xiao et al., 2017, Zhu et al., 2021, Khawar et al., 2020, Liu et al., 2020, Su et al., 2021]. Additionally, modeling all interactions in the same space limits generality and pattern diversity.

Our paper introduces Ad-Rec, a masked transformer-based solution addressing covariate shifts, removing irrelevant cross-features, and modeling diverse feature interactions. Ad-Rec incorporates three key elements: LayerNorm for internal covariate shift mitigation, Multi-head Attention for enhanced generalization, and Attention Masks for eliminating irrelevant cross-features. Ad-Rec

Machine Learning for Systems Workshop at 37th NeurIPS Conference, 2023, New Orleans, LA, USA.



(a) Masked Transformer-based Feature Interaction



(b) Masked Multi-Head Self-Attention

Figure 1: (a) The masked transformer-based feature interaction enables efficient higher-order cross-features by using embedding tables and masking to eliminate irrelevant cross-features. LayerNorm further enhances cross-feature quality. (b) Masked multi-head self-attention employs scalar masks for diverse cross-feature patterns, aided by LayerNorm for faster convergence and improved learning.

effectively captures feature interactions at various orders amidst covariate shifts. In our experiments across diverse non-sequential and sequential models, Ad-Rec achieves the target AUC in 58% training iterations on average, outperforming the state-of-the-art DCN-v2 [Wang et al., 2021]. This results in a training speedup of 1.4× across seven different models and four real-world datasets [CriteoLabs, a,b, Kaggle, Alibaba].

2 Proposed Architecture: Ad-Rec

Ad-Rec uses a masked transformer-based approach (Figure 1a) to handle data drift, reduce noise from irrelevant cross-features, and capture diverse patterns for higher-order interactions. It also incorporates a Bottom MLP ($\mathrm{MLP}_{\mathrm{bot}}$) for dense inputs $\mathbf{x_{dense}}$ and performs embedding for sparse inputs $\mathbf{x_{sparse}}$. The combined output forms the feature sequence $\mathbf{z_0} \in \mathbb{R}^{(N+1)\times D}$ capturing joint embeddings of $N+1$ input features, each associated with a latent vector of size D.

This design empowers Ad-Rec to model feature interactions across multiple dimensions, addressing the challenge of capturing diverse patterns for higher-order cross-features. In deep learning-based recommender systems, the feature sequence $\mathbf{z_0}$ is either used directly or dot-product-based feature interactions are computed as $lower\ triangle(\ \mathbf{z_0} \times \mathbf{z_0^T}\ )$ to extract second-order cross-features. Ad-Rec relies on a masked multi-head attention block to create relevant cross-features while excluding irrelevant ones. This mechanism is visually illustrated in Figure 1b. Each input feature $\mathbf{z_i}$ has query and key-value pairs learned in $\mathbf{h}$ subspaces, corresponding to the number of attention heads. This allows for capturing diverse cross-features in multiple subspaces. This is achieved through linear projection using matrices $\mathbf{W_Q^h}$, $\mathbf{W_K^h}$, and $\mathbf{W_V^h} \in \mathbb{R}^{\mathbf{D}\times\mathbf{D'}}$, where $\mathbf{D'} = \frac{\mathbf{D}}{\mathbf{h}}$. Specifically, the projected query, key, and value vectors for feature $\mathbf{i}$ in subspace $\mathbf{h}$ are denoted as $\mathbf{z_{qi}^h} = \mathbf{z_i W_Q^h}$, $\mathbf{z_{ki}^h} = \mathbf{z_i W_K^h}$, and $\mathbf{z_{vi}^h} = \mathbf{z_i W_V^h}$, respectively. The correlation between feature $\mathbf{i}$ and feature $\mathbf{j}$ under a specific subspace $\mathbf{h}$ is represented by the attention head $\alpha_{\mathbf{i,j}}^{\mathbf{h}}$ (Equation 1), where $\langle\cdot\rangle$ denotes the inner product.

$$\alpha_{\mathbf{i,j}}^{\mathbf{h}} = \frac{\exp\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{kj}^h} \rangle}{\sum_{m=1}^{N+1} \exp\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{km}^h} \rangle} \tag{1}$$

Masking: Ad-Rec uses a masking technique to eliminate irrelevant feature interactions. Each head $\mathbf{h}$ has a mask θ. The masked attention score $\overline{\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{kj}^h} \rangle}$ is calculated as follows (Equation 2):

$$\overline{\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{kj}^h} \rangle} = \begin{cases} \langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{kj}^h} \rangle, & \text{if} \alpha_{\mathbf{i,j}}^{\mathbf{h}} > \theta^{\mathbf{h}} \\ -\infty, & \text{otherwise} \end{cases} \tag{2}$$

The recalculated attention head $\overline{\alpha_{\mathbf{i,j}}^{\mathbf{h}}}$ (Equation 3) is obtained by applying the softmax function to the masked attention scores:

$$\overline{\alpha_{\mathbf{i,j}}^{\mathbf{h}}} = \frac{\exp \overline{\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{kj}^h} \rangle}}{\sum_{m=1}^{N+1} \exp \overline{\langle \mathbf{z_{qi}^h} \cdot \mathbf{z_{km}^h} \rangle}} \tag{3}$$

Finally, the cross-feature of feature $\mathbf{i}$ in subspace $\mathbf{h}$ is updated by combining the relevant feature attentions $\overline{\alpha_{\mathbf{i,j}}^{\mathbf{h}}}$ with the corresponding values $\mathbf{z_v^h}$ (Equation 4):

$$\mathbf{f_i^h} = \sum_{m=1}^{N+1} \overline{\alpha_{i,m}^{h}} \cdot \mathbf{z_{vm}^h} \tag{4}$$

$\theta^{\mathbf{h}}$ is a head-specific mask that is fixed before training. In this work, a geometric sequence of decreasing values, such as $\frac{1}{10^1}, \frac{1}{10^2}, \ldots, \frac{1}{10^h}$, was used for the masks of multiple heads. This choice eliminates irrelevant features. While a single mask could be used for all heads, different masks per head allow for generalization. Appendix D.4 presents an ablation study on different mask values.

LayerNorm: Ad-Rec utilizes Layer Normalization (LayerNorm) to normalize the output of the masked attention layer and feedforward network, ensuring stable and efficient training.

LayerNorm (LN), when applied to $\mathbf{f_i^h}$ of the masked attention layer, is defined using Equation 5.

$$\overline{\mathbf{f_i^h}} = \mathrm{LN}(\mathbf{f_i^h}) = \frac{\mathbf{f_i^h} - \boldsymbol{\mu}_i^{\boldsymbol{h}}}{\sqrt{\boldsymbol{\sigma}_{\boldsymbol{i}}^{\mathbf{2}} + \epsilon}} \odot \boldsymbol{\gamma}^{\boldsymbol{h}} + \boldsymbol{\beta}^{\boldsymbol{h}} \tag{5}$$

Here, $\boldsymbol{\mu}_{\boldsymbol{i}}^{\boldsymbol{h}}$ and $\boldsymbol{\sigma}_{\boldsymbol{i}}^{\mathbf{2}}$ are the mean and variance of $\mathbf{f_i^h}$ across feature dimensions, respectively. The $\odot$ operator represents element-wise multiplication. The learnable parameters $\boldsymbol{\gamma}^{\boldsymbol{h}}$ and $\boldsymbol{\beta}^{\boldsymbol{h}}$ scale and shift the normalized values. The term ϵ ensures numerical stability.

LayerNorm normalizes the output of the masked attention layer, $\mathbf{f_i^h}$, to have zero mean and unit variance across feature dimensions. This mitigates covariate shifts and provides a stable distribution for subsequent layers. The scale and shift parameters, $\boldsymbol{\gamma}^{\boldsymbol{h}}$ and $\boldsymbol{\beta}^{\boldsymbol{h}}$, enable the model to capture appropriate representations for the recommendation task.

LayerNorm also has a similar effect on the output of the feedforward network. By reducing reliance on the scale and distribution of the training dataset, it facilitates faster convergence, offers modest regularization, and improves the efficiency of parameter updates in the recommendation model.

Ad-Rec employs masked multi-head attention (MSA) to learn explicit higher-order cross-features in multiple subspaces, and the feedforward network (FFN) handles implicit interactions. By stacking multiple Ad-Rec layers, up to L (Equations 6 and 7), higher-order interactions can be captured more effectively (see Appendix D.6.1). It also improves performance for larger input feature sequences.

$$\mathbf{z}_{\ell}^{'} = \text{Masked MSA}(\mathrm{LN}(\mathbf{z_{\ell-1}})) + \mathbf{z_{\ell-1}}, \quad \ell = 1, \ldots, \mathbf{L} \tag{6}$$

$$\mathbf{z_\ell} = \mathrm{FFN}(\mathrm{LN}(\mathbf{z}_{\ell}^{'})) + \mathbf{z_\ell}, \quad \ell = 1, \ldots, \mathbf{L} \tag{7}$$

The last layer's output is concatenated with processed dense inputs, resulting in $\mathbf{z}$ (Equation 8).

$$\mathbf{z} = [\mathbf{y_{dense}}; \mathbf{z_\ell}] \tag{8}$$

The final click-through rate (CTR) is obtained by applying the top MLP ($\mathrm{MLP}_{\mathrm{top}}$) to $\mathbf{z}$ (Equation 9). Thus, the problem is modeled as binary classification using binary cross-entropy (BCE) loss to predict whether a user will click the target item.

$$\mathrm{CTR} = \mathrm{MLP}_{\mathrm{top}}(\mathbf{z}) \tag{9}$$

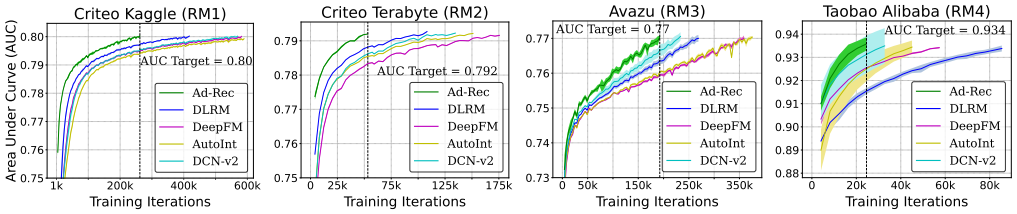


Figure 2: Comparison of Ad-Rec's convergence with baseline feature interaction techniques. The dotted vertical line represents the training iteration where Ad-Rec reaches the target AUC and stops training. On average, Ad-Rec achieves target AUC in 50%, 42%, 45%, and 58% lower iterations, as compared to the DLRM, DeepFM, AutoInt, and DCN-v2 baselines.

3 Experiments and Results

We train recommendation models with varying sizes to represent different classes of at-scale models [Gupta et al., 2020, Adnan et al., 2022a,b]. Their architecture is presented in Table 1 [Wu et al., 2019, Zhao et al., 2019].

Table 1: Recommendation Models Architecture and Ad-Rec Configuration (# layers = 1)

Model	Dataset	Features		Parameters			Neural Network Configuration		Ad-Rec Configuration			Sequential Layer	
		Dense	Sparse	Dense	Sparse	Sparse Dim	Bottom MLP	Top MLP	Num Heads	Hidden Size	FFN Config.	Type	Layers/ Heads
RM1	Criteo Kaggle	13	26	287.5k	33.8M	16	13-512-256-64-16	512-256-1	2	16	128		
RM2	Criteo Terabyte	13	26	549.1k	266M	64	13-512-256-64	512-512-256-1	8	64	512	N/A	N/A
RM3	Avazu	1	21	281.4k	9.3M	16	1-512-256-64-16	512-256-1	2	16	128		
RM4	Taobao Alibaba	1	3	7.3k	5.1M	16	1-16	22-15-15	2	16	128	TSL	1
RM5												MHA	8
RM6												RNN	5
RM7												Transformer	1

Convergence Analysis: We compare the convergence of Ad-Rec with the baseline models (RM1, RM2, RM3, and RM4) by setting a target AUC for each model based on prior work Naumov et al. [2019], Ishkhanov et al. [2020]. The baseline models and Ad-Rec are trained with early stopping to achieve the target AUC. Figure 2 demonstrates that Ad-Rec achieves the target AUC in fewer training iterations, thanks to its masked attention-based feature interaction. On average, Ad-Rec achieves the target AUC in 50%, 42%, 45%, and 58% iterations, surpassing the DLRM, DeepFM, AutoInt, and DCN-v2 baselines. For a comprehensive analysis of Ad-Rec's hyperparameters, please refer to Appendix D.6.

Performance Comparison: Ad-Rec's transformer-based feature interaction is computationally expensive compared to baselines. The wall clock time is measured to compare the runtime of training the recommendation model. As Figure 3 shows, as expected, a single training iteration of Ad-Rec-based training takes more time. Still, it converges in less number of training iterations that provides a speedup of 1.5×, 2×, 2.1× and 1.4× over DLRM, DeepFM, AutoInt, and DCN-v2 baselines.

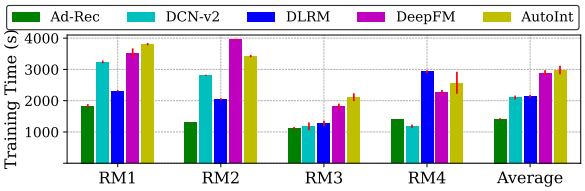


Figure 3: Absolute training time over multiple runs. Ad-Rec converges to target AUC in less time, which translates to a speedup of 1.5×, 2×, 2.1× and 1.4× over DLRM, DeepFM, AutoInt, and DCN-v2 baselines.

4 Conclusion

In this study, we tackled the challenges of covariate shifts and sought to enhance cross-feature learning while accounting for data drift. Our findings underscored the significance of normalizing explicit cross-features and eliminating noisy ones to enhance recommendations in unfamiliar data distributions. To address these challenges, we introduced Ad-Rec, a masked-attention-based feature interaction technique. Through rigorous experimentation, Ad-Rec consistently outperformed state-of-the-art baselines, exhibiting superior quality and convergence speed. It achieved higher AUC scores and accelerated the training process, delivering compelling results across commercial models and publicly available datasets.

References

Muhammad Adnan, Yassaman Ebrahimzadeh Maboud, Divya Mahajan, and Prashant J. Nair. Accelerating recommendation system training by leveraging popular choices. *Proc. VLDB Endow.*, 15(1):127–140, jan 2022a. ISSN 2150-8097. doi: 10.14778/3485450.3485462. URL https://doi.org/10.14778/3485450.3485462.

Muhammad Adnan, Yassaman Ebrahimzadeh Maboud, Divya Mahajan, and Prashant J Nair. Heterogeneous acceleration pipeline for recommendation system training. *arXiv preprint arXiv:2204.05436*, 2022b.

Alibaba. User behavior data from taobao for recommendation. https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1.

Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. Understanding scaling laws for recommendation models, 2022. URL https://arxiv.org/abs/2208.08489.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, DLP-KDD '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367837. doi: 10.1145/3326937.3341261. URL https://doi.org/10.1145/3326937.3341261.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide and deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347952. doi: 10.1145/2988450.2988454. URL https://doi.org/10.1145/2988450.2988454.

CriteoLabs. Criteo display ad challenge, a. https://www.kaggle.com/c/criteo-display-ad-challenge.

CriteoLabs. Terabyte click logs, b. https://labs.criteo.com/2013/12/download-terabyte-click-logs.

Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential / session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 143–153, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3474255. URL https://doi.org/10.1145/3460231.3474255.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

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.

U. Gupta, C. Wu, X. Wang, M. Naumov, B. Reagen, D. Brooks, B. Cottel, K. Hazelwood, M. Hempstead, B. Jia, H. S. Lee, A. Malevich, D. Mudigere, M. Smelyanskiy, L. Xiong, and X. Zhang. The architectural implications of facebook's dnn-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 488–501, 2020. doi: 10.1109/HPCA47549.2020.00047.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569.

Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Proceedings of the 2012th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECMLPKDD'12, page 67–82, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642334856.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

T. Ishkhanov, M. Naumov, X. Chen, Y. Zhu, Y. Zhong, A. G. Azzolini, C. Sun, F. Jiang, A. Malevich, and L. Xiong. Time-based sequence model for personalization and recommendation systems. *CoRR*, abs/2008.11922, 2020. URL https://arxiv.org/abs/2008.11922.

Kaggle. Avazu mobile ads ctr. https://www.kaggle.com/c/avazu-ctr-prediction.

Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenguo Li, and Xiuqiang He. Autofeature: Searching for feature interactions and their architectures for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 625–634, 2020.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 539–548, 2019.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 1754–1763, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220023. URL https://doi.org/10.1145/3219819.3220023.

Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2636–2645, 2020.

MLCommons. Mlperf benchmarks. https://mlcommons.org/en/training-normal-10/.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019. URL https://arxiv.org/abs/1906.00091.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 111–112, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334730. doi: 10.1145/2740908.2742726. URL https://doi.org/10.1145/2740908.2742726.

Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 255–262, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939704. URL https://doi.org/10.1145/2939672.2939704.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.

Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. Detecting beneficial feature interactions for recommender systems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4357–4365, 2021.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pages 1785–1797, 2021.

Markus Weimer, Alexandros Karatzoglou, Quoc Le, and Alex Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/f76a89f0cb91bc419542ce9fa43902dc-Paper.pdf.

C. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344, Feb 2019. doi: 10.1109/HPCA.2019.00048.

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. *arXiv preprint arXiv:1708.04617*, 2017.

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. Recommending what video to watch next: A multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 43–51, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi: 10.1145/3298689.3346997. URL https://doi.org/10.1145/3298689.3346997.

Chenxu Zhu, Bo Chen, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Aim: Automatic interaction machine for click-through rate prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

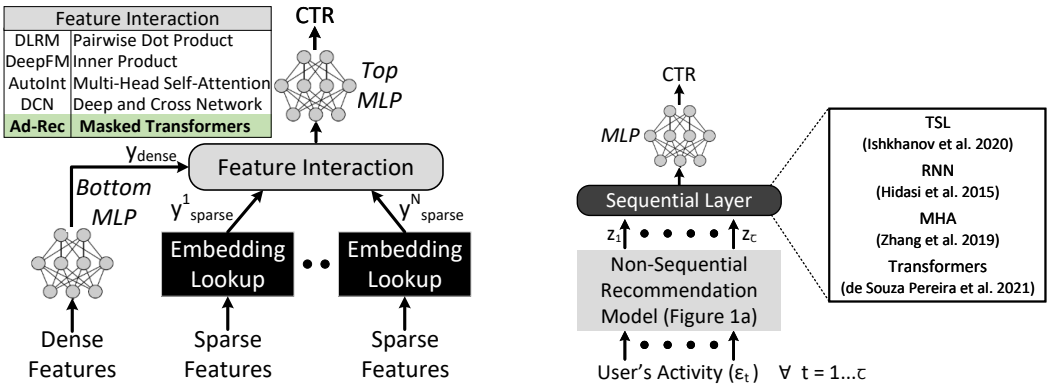
A Appendix

A.1 High-Level Overview: Deep Learning-based Recommendations with Ad-Rec

Figure 4a illustrates the model architecture of Ad-Rec, a non-sequential deep learning-based recommendation model inspired by DLRM [Naumov et al., 2019]. The model takes two types of inputs: dense and sparse features. Dense inputs consist of continuous features such as the user's age or the time of day. Sparse inputs include categorical features like the user's location or liked videos.

Multi-layer perceptrons (MLPs) are employed to process the dense inputs, while large embedding tables handle the sparse inputs, each representing a specific categorical feature. These embeddings, along with the dense features, are then passed through a feature interaction layer, enabling the generation of cross-features that capture complex relationships between different features. Next, the cross-features and dense features are fed into an MLP layer, which predicts the click-through rate (CTR). The CTR indicates the likelihood of a user clicking on an item, serving as a key metric for recommendation models. By leveraging the power of deep learning and effective feature interaction, Ad-Rec enhances the accuracy and performance of recommendation systems.

Figure 4b provides an overview of sequential recommendation models within the context of Ad-Rec. These models explicitly incorporate the temporal aspect of user-item interactions through a set of events denoted as ε. Each event ε represents a user u interacting with an item i at a specific time t.



(a) Non-Sequential Recommendation Model

(b) Sequential Recommendation Model

Figure 4: (a) Non-Sequential Deep Learning based Recommendation Model with multiple feature interaction techniques. (b) Sequential recommendation model with multiple sequential layers.

User behaviour is captured as a sequence of events ε spanning multiple time steps, denoted as $t = \{1, \ldots, \tau - 1\}$. The sequential recommendation model is then trained to predict the next event at time step τ, based on the previous event sequence. This shift towards sequential modelling transforms the organization of the training dataset, as it now represents a sequence of user activities rather than a simple collection of user interactions with features, as seen in non-sequential recommendation models. By incorporating temporal dynamics, Ad-Rec enables more accurate and personalized recommendations in dynamic user environments.

A.2 Application to Sequential Recommendation Models

For sequential recommendation models, we apply Ad-Rec by generating an embedding vector $\mathbf{z}_t$ for each event ε_t using the non-sequential recommendation model. Here, ε_t represents the event at time step t, such as a user click or purchase. These embeddings capture event characteristics, including explicit timing as a dense feature. The width of the last layer in the non-sequential model is adjusted to match the user-interaction vector's width, which is then fed into the sequential layer. This produces

a sequence of embedding vectors Z (Equation 11).

$$\mathbf{z}_t = \mathbf{AdRec}(\varepsilon_t) \qquad t = 1, \ldots, \tau \tag{10}$$

$$Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\tau-1}] \tag{11}$$

$$\mathbf{c} = \mathbf{SequentialLayer}(\mathbf{z}_\tau, Z) \tag{12}$$

Ad-Rec generates an embedding $\mathbf{z}_t$ for each event, capturing relevant information about the user-item interaction. These embeddings are used as input to the sequential layer, which considers the context and temporal order of earlier events to predict the next event ε_τ at time step τ (Equation 12). By leveraging the Ad-Rec embeddings, the model effectively captures user-item interactions within the sequence, leading to improved recommendation performance.

A.3 Multihead Self-Attention

The standard self-attention mechanism, commonly used in Natural Language Processing (NLP) tasks, forms a fundamental building block. Given an input sequence $\mathbf{z} \in \mathbb{R}^{N \times D}$, it computes a weighted sum over all values $\mathbf{v}$ in the sequence. The attention weights $\mathrm{A_{ij}}$ reflect the pairwise similarity between query $\mathbf{q_i}$ and key $\mathbf{v_j}$ representations of the input tokens.

$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathbf{z}\mathbf{U}_{qkv} \qquad \mathbf{U}_{qkv} \in \mathbb{R}^{D \times 3D} \tag{13}$$

$$\mathbf{A} = \mathrm{softmax}(\frac{\mathbf{q}\mathbf{k}^\mathbf{T}}{\sqrt{\mathbf{D}}}) \qquad \mathbf{A} \in \mathbb{R}^{N \times N} \tag{14}$$

$$\mathrm{SA} = \mathbf{A}\mathbf{v} \tag{15}$$

Multihead self-attention (MSA) extends the self-attention mechanism by performing H parallel self-attention operations, where each self-attention is referred to as a "head". The parameter H is a hyper-parameter that determines the number of heads. To ensure constant computation and parameter count with respect to H, the dimension D (Eq. 16) is set to $\frac{D}{H}$.

$$\mathrm{MSA}(\mathbf{z}) = [\mathrm{SA}_1(\mathbf{z})\ ; \mathrm{SA}_2(\mathbf{z})\ ; \ldots\ ; \mathrm{SA}_n(\mathbf{z})]\mathbf{U}_{msa} \qquad \mathbf{U}_{msa} \in \mathbb{R}^{HD \times D} \tag{16}$$

A.4 Evaluation Setup

We compare Ad-Rec against four state-of-the-art techniques. DLRM [Naumov et al., 2019] and DeepFM [Guo et al., 2017] are 2^{nd} order feature interaction while AutoInt [Song et al., 2019] and DCN-v2 [Wang et al., 2021] are higher-order feature interaction techniques.

We investigate Ad-Rec's performance using various real-world datasets for both non-sequential and sequential recommendation tasks. For non-sequential tasks, we employ three datasets: Criteo Kaggle [CriteoLabs, a], Criteo Terabyte [CriteoLabs, b], and Avazu [Kaggle]. For sequential recommendation models, we turn to the Taobao User Behavior dataset [Alibaba].

Training Details: We used PyTorch-1.9 for our experiments, building on DLRM [Naumov et al., 2019] for non-sequential recommendations and TBSM [Ishkhanov et al., 2020] for sequential ones. All models were implemented identically except for the feature interaction component.

Non-sequential models (RM1, RM2, and RM3) used SGD [Bottou, 2012] with batch sizes of 128 (RM1 and RM3) and 1024 (RM2). Learning rates were 0.01 (RM1), 0.1 (RM2), and 0.2 (RM3). Sequential models (RM4, RM5, RM6, and RM7) employed Adagrad [Duchi et al., 2011] with a learning rate of 0.05 and a batch size of 128. We conducted five independent runs and reported mean and standard deviation results.

Evaluation Metrics: For evaluating the performance of our recommendation models, we use the Area Under Curve (AUC) metric, as established by the MLPerf community [MLCommons, Cheng et al., 2016, Guo et al., 2017, Li et al., 2019]. We also track testing accuracy and BCE loss.

B Related Work

Enhanced Models: Recommendation models, crucial for personalized experiences, often face limitations in capturing complex feature interactions using traditional methods like collaborative filtering (CF) and matrix factorization (MF) [He et al., 2017, Koren et al., 2009, Weimer et al., 2007, Hidasi and Tikk, 2012]. Deep learning models, including MLP, neural networks, autoencoders, and GRU, improve feature interaction modeling [Cheng et al., 2016, Zhao et al., 2019, Naumov et al., 2019, Sedhain et al., 2015, Hidasi et al., 2015]. While sequential and session-based models address temporal aspects [Ishkhanov et al., 2020, Chen et al., 2019, Sun et al., 2019, de Souza Pereira Moreira et al., 2021], challenges in capturing intricate feature relationships remain.

Learning Feature Interactions: Recommendation models have evolved to better capture feature interactions, with deep learning methods such as MLP, neural networks, autoencoders, and GRU [Cheng et al., 2016, Zhao et al., 2019, Naumov et al., 2019, Sedhain et al., 2015, Hidasi et al., 2015]. However, existing approaches primarily focus on lower-order interactions, struggling to capture higher-order feature interactions. Traditional methods like CF and MF [He et al., 2017, Koren et al., 2009, Weimer et al., 2007, Hidasi and Tikk, 2012] also have limitations in this regard.

Eliminating Useless Features: Existing methods in recommendation systems aim to enhance feature interaction modeling. AFM differentiates between feature interactions but fails to eliminate cross-features [Xiao et al., 2017]. AIM and AutoFIS use selection gating to remove irrelevant interactions [Liu et al., 2020], and AutoFeature identifies essential interactions with NAS-based techniques [Khawar et al., 2020]. These methods often address specific aspects and isn't comprehensive.

C Ad-Rec: Visual Intuition

To analyze the feature interaction of DLRM and Ad-Rec, we randomly sampled a test input from the real-world Taobao user behaviour dataset [Alibaba]. This input consists of sequential user activity with a length of 21, a timestamp as a dense feature, and sparse features for user, item, and category. Notably, the ground truth of the sampled input indicates that it is a **negative sample**.

Figure 5 presents the cosine similarity heat map, comparing the feature interaction of DLRM and Ad-Rec. In DLRM, the feature interaction is based on dot product calculations, while Ad-Rec employs masked attention. The features involved in the interaction include the user's activity timestamp (feature 1), user ID (feature 2), item ID (feature 3), and item category (feature 4).

Examining the sequential interaction at timestamp $t = 2$ in Figure 5, we observe contrasting behaviour between DLRM and Ad-Rec. We observe that DLRM's dot product-based feature interaction provides close similarity across all pairs. Contrary to this, Ad-Rec's masked attention-based feature interaction considers Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) projections. Applying a mask with $\frac{1}{10^3}$ value reveals no features being masked (note that the mappings of attention heads can be found in Appendix D.7).

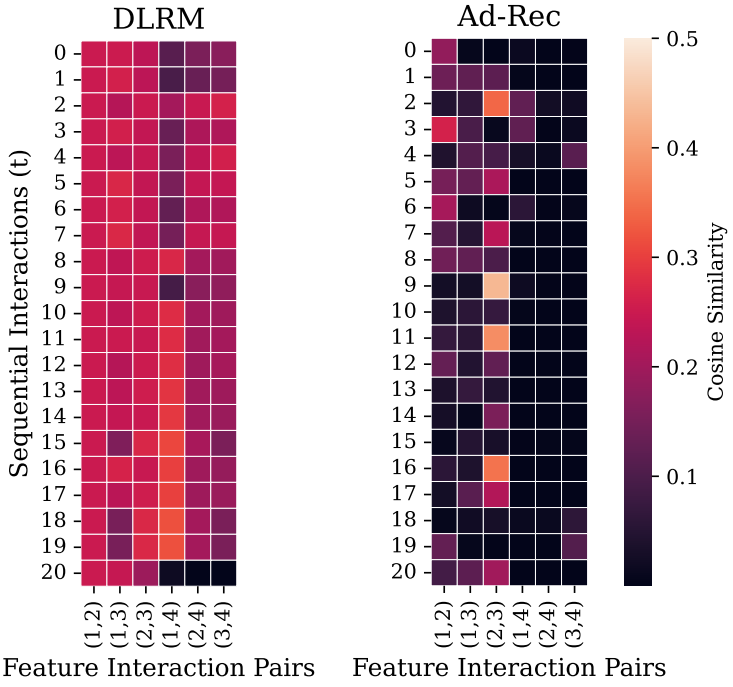


Figure 5: Cosine similarity of feature interaction in the real-world Taobao dataset: one dense and three sparse features for a single user.

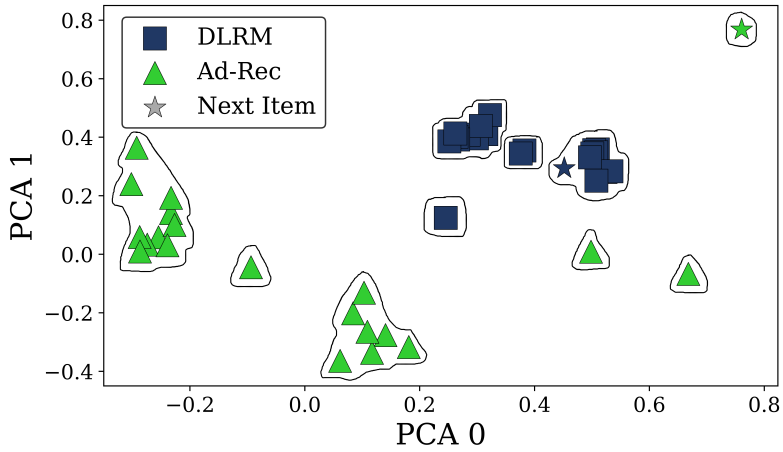


Figure 6: Principal Component Analysis (PCA) to represent high-dimensional sequential user activity in 2-dimensional space. This compares DLRM and Ad-Rec generated embedding vectors for the RM4 model. Note that the next item (denoted by $\star$) is a *negative sample*. Thus, a *large Euclidean distances* between the next item and sequential interaction clusters indicate a higher quality model.

The highest weight is assigned to feature pair $(2, 3)$, indicating a strong correlation between user ID and item ID. In contrast, other pairs such as $(1, 2)$, $(1, 4)$, $(2, 4)$, and $(3, 4)$ show weak alignment, implying a primarily **negative sample**. Unlike DLRM's dot product-based feature interaction, Ad-Rec captures this information.

C.1 Sequential Embedding Vectors' Feature Interaction

The quality of sequence embedding vectors (Z) plays a crucial role in the prediction quality of sequential recommendation. To better understand the patterns within these vectors, we utilize Principal Component Analysis (PCA) to map the embedding vectors (with a width of 16) into a 2-dimensional space. In this space, similar interactions are grouped closely together, while dissimilar interactions are distanced apart. The proximity of the next item's placement to historical item interactions indicates a higher probability of the user clicking on that item. Figure 6 depicts the PCA plot for sequential embedding vectors generated by both feature interaction methods, as explained in Section C. Each method generates 21 embedding vectors, representing the user's historical interactions, with the next item shown by a star symbol $(\star)$.

In the plot, closely related interactions form clusters, and the relative Euclidean distance between clusters signifies their correlation. For DLRM-based feature interaction, we observe that all sequential interactions are clustered together, resulting in four closely located clusters in the Euclidean space. On the other hand, Ad-Rec-based feature interaction generates more distinct clusters based on the type of user interactions. The sequential layer predicts the likelihood of the user clicking on the next item vector based on the placement of the next item embedding in relation to the user's sequential embedding vectors.

Interestingly, we notice that Ad-Rec-based feature interaction generates an embedding vector for the next item that is situated far away from previous user interactions, indicating that the next item does not have a strong connection to the user's sequential interactions. Conversely, for DLRM-based feature interaction, the next item is located within the cluster, suggesting a close relationship with most previous interactions. The placement of Ad-Rec-generated sequence vectors and subsequent item vectors aligns with the ground truth, as the input represents a negative sample.

D Ablation Studies

To understand the effect of different components of Ad-Rec on the overall quality of recommendation models, we conducted multiple ablation studies.

D.1 Positional Embedding

While positional embeddings are commonly used in language tasks to preserve word order, they are not utilized in Ad-Rec for recommender systems. Despite this, we evaluated 1-D positional embeddings to encode spatial information of features.

In our sensitivity study, we explored different ways of encoding the spatial information of features using positional embeddings inspired by NLP models. We considered two cases:

- **No positional information:** This case involved using only feature embeddings and providing them as-is to the transformer encoder. This approach was the default across all other experiments in the paper.
- **1-dimensional positional embedding:** We treated the input features as a sequence of features, assigning each sparse feature and dense feature vector a position based on the embedding table position for language models. We added position embeddings to the feature inputs just before feeding them to the Transformer encoder. The dimension of the position embedding was kept similar to the sparse feature dimension, and the number of position embeddings was equal to the number of features.

Equation 17 demonstrates the incorporation of positional embeddings into the input features. The feature inputs and their respective embeddings were summed with the position embeddings.

$$\mathbf{z_0} = [\text{MLP}(\mathbf{x_{dense}}); \mathbf{x^1_{sparse}}\mathbf{E^1}; \ \ldots\ ; \ \mathbf{x^N_{sparse}}\mathbf{E^N}\ ] + \mathbf{E_{pos}} \quad \mathbf{E} \in \mathbb{R}^{M\times D},\ \mathbf{E_{pos}} \in \mathbb{R}^{(N+1)\times D} \tag{17}$$

Table 2 presents the evaluation results comparing the models with and without positional embeddings. Our hypothesis was that since the Multi-Head Attention block exhibits permutation-equivariance, the position of a feature in the input sequence does not encode useful information. Therefore, the models that directly input the raw features to the masked transformer encoder, facilitating higher-order feature interaction, performed the best. Incorporating 1-dimensional positional embeddings led to a degradation in model performance, even worse than the DLRM baseline. Notably, in certain models like RM3, incorporating 1-dimensional positional embeddings prevented the model from converging.

Table 2: Results of ablation study on positional embeddings with single epoch training.

Model	AUC		Test Accuracy (%)		BCE Loss	
	No Pos. Emb.	1-D Pos. Emb.	No Pos. Emb.	1-D Pos. Emb.	No Pos. Emb.	1-D Pos. Emb.
RM1	**0.801**	0.796	**78.70**	78.34	**0.455**	0.461
RM2	**0.790**	0.786	**81.22**	80.94	**0.423**	0.426
RM3	**0.775**	Not Converge	**83.76**	Not Converge	**0.382**	Not Converge

Figure 7 illustrates that recommendation models achieve convergence in significantly fewer training iterations when no spatial information is added to the input feature embeddings. Even the model with 1-dimensional positional embedding requires more training iterations than the baseline DLRM with second-order cross-features to reach the target AUC.

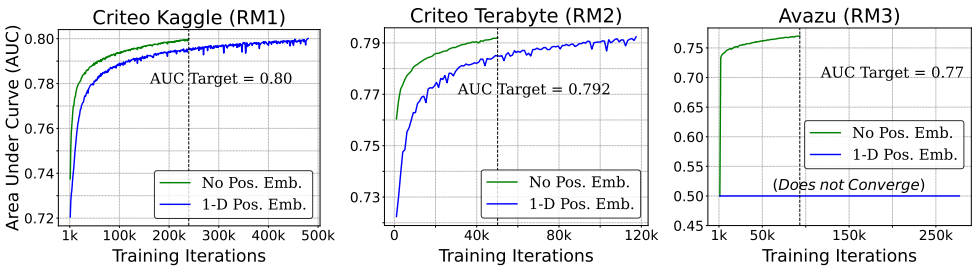


Figure 7: Model convergence with and without positional embedding.

Table 3: Comparing the impact of LayerNorm (LN) in Ad-Rec to handle covariate shift.

Model	AUC		BCE Loss	
	Ad-Rec w/ LN	Ad-Rec w/o LN	Ad-Rec w/ LN	Ad-Rec w/o LN
RM1	**0.801 (1e-4)**	0.795 (2.4e-4)	**0.455 (1.4e-3)**	0.460 (1.4e-3)
RM2	**0.790 (1.8e-4)**	0.785 (2.2e-4)	**0.423 (9.4e-5)**	0.426 (1.4e-4)
RM3	**0.775 (3.2e-4)**	0.771 (1.8e-4)	**0.382 (1.2e-4)**	0.385 (1.4e-4)
RM4	**0.949 (1.9e-3)**	0.944 (2.8e-3)	**0.232 (8.3e-3)**	0.247 (7.4e-3)
RM5	**0.895 (2.1e-3)**	0.890 (2.8e-3)	**0.361 (2e-3)**	0.366 (5.9e-3)
RM6	**0.852 (2.6e-3)**	0.823 (7.3e-3)	**0.400 (4.2e-4)**	0.427 (6.3e-3)
RM7	**0.940 (1.8e-3)**	0.930 (2e-3)	**0.272 (5.6e-3)**	0.284 (4.8e-3)

D.2 Ablation Study for Covariate Shift

We evaluate the importance of LayerNorm in Ad-Rec with an ablation study that removes the LayerNorm layer while keeping the rest of the architecture unchanged. Table 3 shows a decrease in the AUC metric and an increase in the BCE Loss when LayerNorm is omitted. Thus LayerNorm is key to handle data distribution drift, promoting faster, and more stable convergence in Ad-Rec.

D.3 Computational Cost Analysis

To compare the computational cost of Ad-Rec-based feature interaction, we trained DLRM and Ad-Rec using a fixed computational budget and compared the training quality metric (AUC). Figure 8 shows that Ad-Rec dominates DLRM on this performance-compute trade-off. Similar trends are observed for other remaining models (RM2 and RM4).

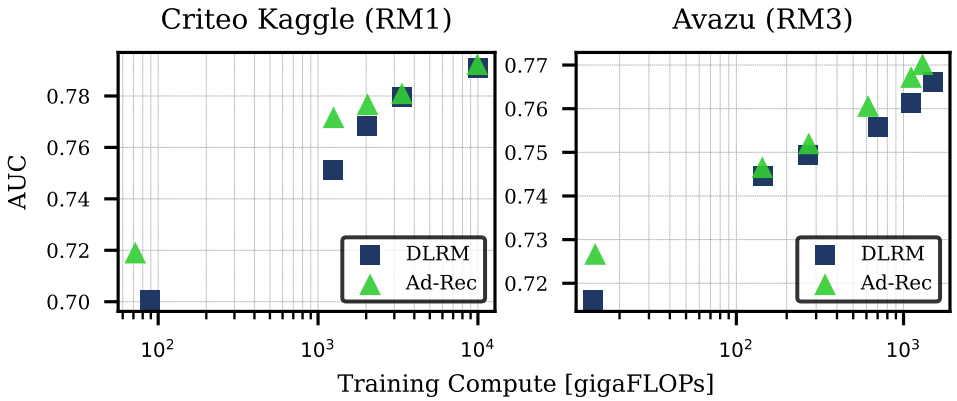


Figure 8: Model quality versus computational cost for different models. Ad-Rec outperforms state-of-the-art DLRM with the same computational budget.

D.4 Mask Analysis

In our ablation study, we examined the impact of masking and the choice between a fixed mask value or different mask values for each attention head. Three scenarios were considered: no mask, a fixed mask value for all heads, and different mask values for each head. Table 4 illustrates the advantages of employing distinct mask values for each head, highlighting the superiority over using a fixed mask value or no masking.

Models trained with masking generally demonstrate improved AUC compared to models without masking, as masking effectively eliminates irrelevant cross-features that can degrade prediction quality. However, determining the optimal mask threshold presents a challenge. Different models require different mask thresholds (θ), and selecting an unsuitable mask can harm prediction quality by eliminating important features. Prior approaches [Liu et al., 2020, Khawar et al., 2020, Zhu et al., 2021] tackle this issue by training models specifically to learn such features, but this approach incurs computational costs and lacks generalizability across models or even the same model with

Table 4: Ablation Study. Mask (θ) Analysis with Single Epoch Training - AUC Mean (std deviation). The 'No Mask' entry indicates a scenario containing no masking. The numbers in other entries show fixed mask values across all heads, and Ad-Rec employs different mask values across each head.

Model	Mask Value (θ)						
	No Mask	0.1	0.01	0.05	0.001	0.005	Ad-Rec
RM1	0.801095 (3.17e-4)	0.801269 (3.74e-4)	0.801357 (2.2e-4)	0.801055 (4.33e-4)	0.801112 (4.82e-4)	0.801199 (1.14e-4)	**0.80153 (1.09e-4)**
RM2	0.790164 (2.44e-4)	0.790222 (1.84e-4)	0.790214 (1.92e-4)	0.790176 (8.07e-5)	0.790185 (4.56e-4)	0.790179 (1.71e-4)	**0.790224 (1.56e-4)**
RM3	0.775091 (4.85e-4)	0.775081 (7.9e-4)	0.775044 (4.08e-4)	0.775344 (9.2e-4)	0.775637 (2.1e-4)	0.77587 (3.21e-4)	**0.775971 (4.3e-4)**
RM4	0.9453581 (2.88e-3)	0.9421871 (6.46e-3)	0.9465051 (2.57e-3)	0.9430625 (3.67e-3)	0.9434243 (4.54e-3)	0.9406334 (1.45e-3)	**0.9494345 (1.9e-3)**
RM5	0.8940437 (2.87e-4)	0.8938943 (2.93e-4)	0.89249 (2.04e-3)	0.8866167 (9.16e-3)	0.8882817 (5.43e-3)	0.8892716 (4.01e-3)	**0.8955358 (7.65e-3)**
RM6	0.8500284 (1.73e-3)	0.840935 (8.25e-3)	0.8453409 (6.09e-3)	0.8424109 (6.67e-3)	0.8491047 (5.64e-3)	0.8374637 (4.19e-3)	**0.8520705 (2.61e-3)**
RM7	0.9323605 (1.45e-3)	0.9319719 (5.68e-3)	0.9326281 (1.05e-3)	0.9337075 (1.84e-3)	0.9317426 (8.62e-3)	0.9327804 (1.45e-3)	**0.9404412 (1.82e-3)**

different sizes. In contrast, Ad-Rec addresses this challenge by utilizing different mask values for each attention head, facilitating better generalization. If an important feature is eliminated in one head, it can be compensated for by other attention heads, resulting in a higher-quality model.

D.5 Evaluation Metrics for a Training Epoch

Table 5 compares evaluation metrics for a single training epoch. Across all models and datasets, Ad-Rec consistently outperforms the baselines regarding AUC. On average, Ad-Rec improves the AUC metric by 0.012, 0.008, 0.006, and 0.001 compared to the DLRM, DeepFM, AutoInt, and DCN-v2 baselines. Although higher-order feature interaction techniques may exhibit lower performance in certain models and datasets, Ad-Rec showcases superior generalization capabilities. It consistently outperforms other feature interaction techniques.

Table 5: Evaluation Metric Comparison with Single Epoch Training - Mean (stddev)

Model	AUC					BCE Loss				
	DLRM	DeepFM	AutoInt	DCN-v2	Ad-Rec	DLRM	DeepFM	AutoInt	DCN-v2	Ad-Rec
RM1	0.798 (1.9e-4)	0.796 (1.6e-4)	0.795 (8.12e-5)	0.796 (2.9e-4)	**0.801 (1e-4)**	0.459 (1.5e-3)	0.461 (1.6e-3)	0.461 (1.4e-3)	0.460 (1.3e-3)	**0.455 (1.4e-3)**
RM2	0.788 (1.8e-4)	0.783 (1.5e-4)	0.785 (1.5e-4)	0.786 (8.29e-5)	**0.790 (1.8e-4)**	0.424 (1.4e-4)	0.428 (1.5e-4)	0.426 (7.25e-5)	0.426 (5.14e-5)	**0.423 (9.4e-5)**
RM3	0.768 (9.5e-4)	0.763 (3.4e-3)	0.763 (7.8e-4)	0.772 (1e-3)	**0.775 (3.2e-4)**	0.386 (3.8e-4)	0.390 (2.6e-4)	0.390 (3.6e-4)	0.384 (5e-4)	**0.382 (1.2e-4)**
RM4	0.933 (1.3e-4)	0.939 (6.6e-4)	0.942 (1.5e-3)	0.947 (5.8e-3)	**0.949 (1.9e-3)**	0.267 (3.4e-3)	0.257 (1.3e-3)	0.254 (3.8e-3)	0.237 (1.5e-2)	**0.232 (8.3e-3)**
RM5	0.869 (4.2e-4)	0.881 (1.1e-4)	0.893 (5.5e-3)	0.894 (1.5e-3)	**0.895 (2.1e-3)**	0.378 (1.8e-4)	0.360 (6.3e-3)	0.363 (1.9e-3)	0.361 (3.2e-3)	**0.361 (2e-3)**
RM6	0.840 (7.1e-3)	0.850 (1e-3)	0.849 (2.8e-3)	**0.855 (1.3e-3)**	0.852 (2.6e-3)	0.385 (1.4e-3)	0.388 (3.7e-3)	0.388 (7.2e-3)	**0.375 (1.1e-3)**	0.380 (4.2e-3)
RM7	0.920 (1.8e-3)	0.934 (1.3e-3)	0.932 (6e-3)	**0.940 (5.2e-3)**	**0.940 (1.8e-3)**	0.299 (1.1e-3)	0.269 (2.8e-3)	0.279 (1.2e-3)	0.255 (1.3e-3)	**0.254 (5.6e-3)**

D.6 Ad-Rec: Masked Transformer Hyper-parameters Analysis

To evaluate the robustness of our Ad-Rec model, we conducted an extensive study by varying the hyperparameters of the transformer encoder. Table 6 provides an overview of the different hyperparameters considered in this sensitivity analysis.

Table 6: Scaling hyper-parameters of transformer encoder.

Num Layers N	Num Heads H	Dropout Ratio D	Non-Linear Activation A	Hidden Size d_{model}	Linear Config. d_{ff}
1	1	0.01	ReLU	16	128
2	2	0.05	GeLU	64	512
4	4	0.1			
	8	0.2			
	16	0.3			

D.6.1 Number of Layers

We investigated the impact of the number of masked transformer layers (N) in the RM1 model while keeping other hyperparameters constant. Figure 9 showcases the test accuracy and BCE loss for the RM1 model. Our observations revealed that increasing the number of masked transformer layers did not yield significant benefits due to the smaller sequence length of RM1 (27 features). However, we anticipate that models with a larger number of features and longer sequence lengths would demonstrate improved AUC with more masked transformer layers. This is because higher-order feature interaction becomes more valuable when there are more features. In large-scale industrial datasets, the number of sparse features can reach thousands. For instance, the Meta synthetic dataset[1], which cannot be used for training due to its synthetic nature, contains 856 sparse features. Nevertheless, it highlights the scale at which Ad-Rec can significantly enhance prediction accuracy.

[1] https://github.com/facebookresearch/dlrm_datasets

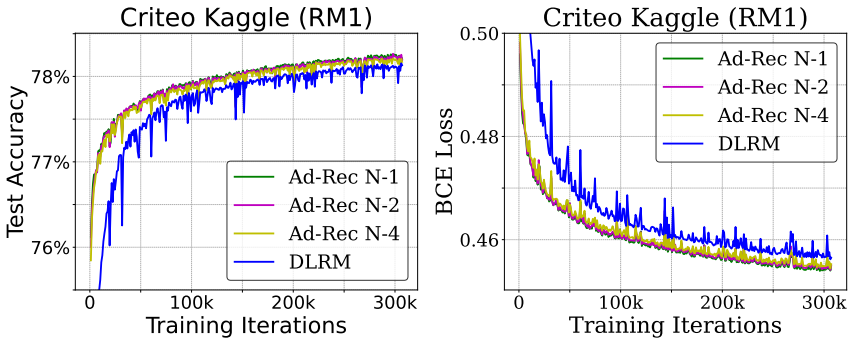


Figure 9: Test Accuracy and BCE Loss with varying layers of Ad-Rec masked transformer.

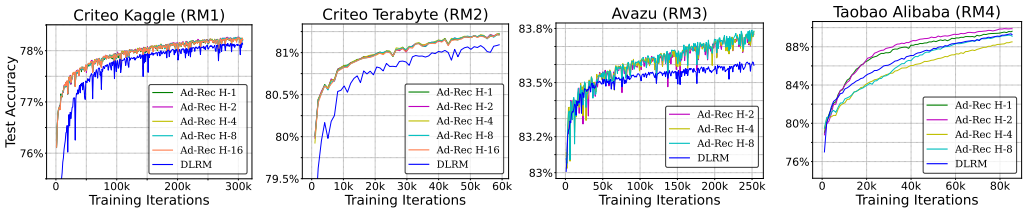


Figure 10: Trends in test accuracy by varying the number of attention heads of masked transformer for RM1, RM2, RM3, and RM4 models. Models with missing head (RM3 and RM4 with 16 heads) means the model could not converge.

D.6.2 Number of Attention Heads

We employ masked multi-head attention to enable higher-order feature interactions across multiple subspaces. This technique divides a single embedding vector into multiple heads of the same length, allowing for parallel execution and feature interaction within different subspaces. The number of attention heads (H) varies from 1 to 16 for each model (RM1, RM2, RM3, and RM4). The relationship between the test accuracy and the number of attention heads is illustrated in Figure 10. Notably, the computational complexity remains unchanged as all the heads are concatenated at the end, and each head operates on a portion of the embedding vector.

Our observations indicate that models with an intermediate number of heads consistently converge and yield higher accuracy. In contrast, the RM3 model fails to converge when using a single-head model, emphasizing the importance of feature interaction in multiple subspaces. Furthermore, when the number of heads equals the length of the embedding vector D (i.e., $H = D$), the feature interaction becomes excessively fine-grained, leading to model divergence. For the RM3 and RM4 models, it was observed that neither model converged when using $(H = 16)$ and $(D = 16)$, reinforcing the need for an appropriate balance in the number of attention heads to achieve optimal performance.

D.6.3 Dropout Ratio

The masked transformer architecture incorporates a residual connection, depicted in Figure 1a, to ensure effective information flow and gradient propagation. Additionally, a dropout mechanism prevents overfitting by randomly replacing input features with random features. To investigate the impact of the dropout ratio on model predictions, we varied the ratio across all models, ranging from 0.01 to 0.3.

Figure 11 showcases the relationship between the dropout ratio and test accuracy. Our findings consistently demonstrate that lower dropout values (0.01 - 0.05) yield superior predictions across all models. These smaller dropout ratios enable cross-features to incorporate with the original raw features, facilitating improved learning of implicit interactions. As the dropout ratio increases, the test accuracy gradually declines, eventually approaching the performance of the baseline DLRM model when the dropout ratio reaches 0.3.

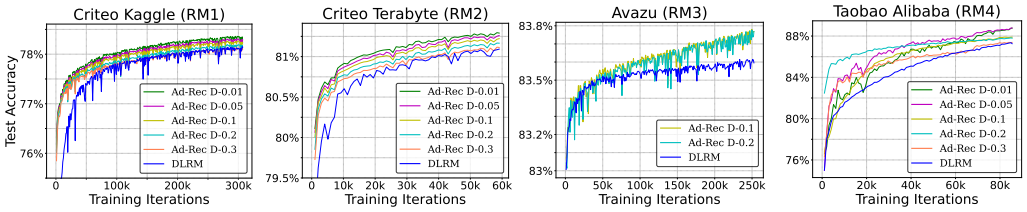


Figure 11: Trends in test accuracy by varying dropout ratio of residual connection in masked transformer for RM1, RM2, RM3, and RM4 model. Models with missing dropout values (RM3 with a dropout of 0.01 and 0.05) mean the model could not converge.

The diminishing test accuracy with higher dropout ratios can be attributed to introducing more random features into the cross-features. These additional random features may disrupt the underlying patterns and relationships in the data, negatively impacting model performance. This finding aligns with previous research [Song et al., 2019], which utilizes raw features without dropout in multi-head attention-based cross-features to learn higher-order interactions. However, in the case of Ad-Rec, a smaller dropout ratio is employed during training to enhance generalization, while dropout is removed during inference for optimal performance.

D.6.4 Non-Linear Activation

Figure 12 presents the investigation into the impact of non-linear activation functions on test accuracy. Surprisingly, transitioning from the default ReLU activation to GeLU activation does not yield any noticeable effect on the test accuracy of the models. Regardless of the chosen non-linear activation function, all models across different datasets converge to the same point.

This finding suggests that the specific type of non-linearity employed in the activation function does not significantly influence the extraction of cross-features. It indicates that the masked transformer architecture is robust to different non-linearities, and the models can effectively capture and learn the underlying interactions between features regardless of the specific activation function used.

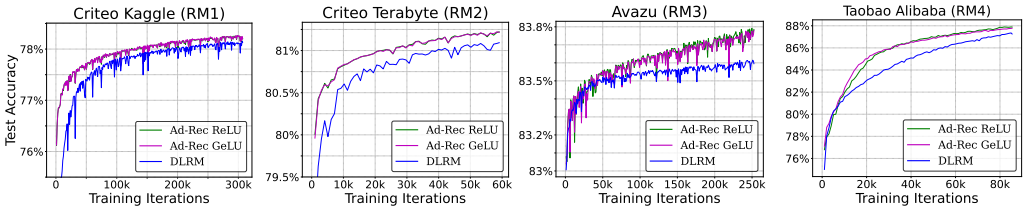


Figure 12: Trends in test accuracy by changing the non-linear activation of a feed-forward network of the masked transformer. It does not have any effect on the model quality.

D.7 Scaling laws of Recommendation Models

We conducted scaling ablation studies on DLRM-style recommendation models. Scaling the embedding dimension had the most significant improvement while scaling the model size had minimal effect on model quality. In the ablation studies with Ad-Rec, we evaluated model quality using masked-attention-based feature interaction while keeping the model size and computational budget fixed. Table 7 provides an overview of the scaled components and their corresponding configurations, along with the model size in terms of parameters. This analysis allows us to assess the impact of scaling on model quality. Previous research [Ardalani et al., 2022] has explored the scaling laws for non-sequential recommendation models, particularly focusing on Click-Through Rate (CTR) in DLRM-style models. Their findings revealed that increasing the model size did not significantly enhance accuracy, while training on more data led to slight improvements. In contrast, Figure 13 demonstrates the behaviour of model loss as different components, including the Ad-Rec loss, are scaled. Interestingly, even with fewer parameters (approximately half), Ad-Rec outperforms other

models regarding convergence speed. This emphasizes the significance of higher-order feature interaction, eliminating irrelevant features, and addressing covariate shifts in improving the representation of input features. Merely scaling existing components or increasing the size of training datasets does not yield comparable results. The success of Ad-Rec opens up new avenues for research in recommendation model architecture.

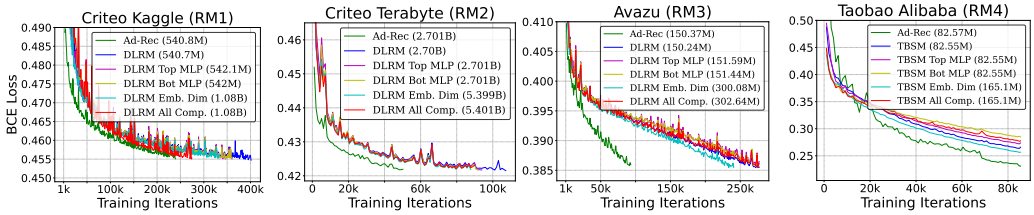


Figure 13: Recommender models were studied to understand the impact of component scaling on model quality. Despite a small training dataset, Ad-Rec outperformed these scaled models.

Table 7: Scaling Recommendation Models Components

Model	Scaling Component	Sparse Dimension	Neural Network Configuration		Model Parameters
			Bottom MLP	Top MLP	
RM1	(N/A) DLRM	16	13-512-256-64-16	512-256-1	540.7M
RM1	(N/A) Ad-Rec	16	13-512-256-64-16	512-256-1	540.8M
RM1	Sparse Emb. Dim	32	13-512-256-64-32	512-256-1	1.08B
RM1	Top MLP	16	13-512-256-64-16	1024-768-512-256-1	542.1M
RM1	Bottom MLP	16	13-1024-768-512-256-128-64-16	512-256-1	542M
RM1	All Comp.	32	13-1024-768-512-256-128-64-32	1024-768-512-256-1	1.08B
RM2	(N/A) DLRM	64	13-512-256-64	512-512-256-1	2.7B
RM2	(N/A) Ad-Rec	64	13-512-256-64	512-512-256-1	2.701B
RM2	Sparse Emb. Dim	128	13-512-256-128	512-512-256-1	5.399B
RM2	Top MLP	64	13-512-256-64	1024-768-512-512-256-128-1	2.701B
RM2	Bottom MLP	64	13-1024-768-512-256-128-64	512-512-256-1	2.701B
RM2	All Comp.	128	13-1024-768-512-256-128	1024-768-512-512-256-128-1	5.401B
RM3	(N/A) DLRM	16	1-512-256-64-16	512-256-1	150.24M
RM3	(N/A) Ad-Rec	16	1-512-256-64-16	512-256-1	150.37M
RM3	Sparse Emb. Dim	32	1-512-256-64-32	512-256-1	300.08M
RM3	Top MLP	16	1-512-256-64-16	1024-768-512-256-128-64-1	151.59M
RM3	Bottom MLP	16	1-1024-768-512-256-128-64-16	512-256-1	151.44M
RM3	All Comp.	32	1-1024-768-512-256-128-64-32	1024-768-512-256-128-64-1	302.64M
RM4	(N/A) DLRM	16	1-16	15-15	82.55M
RM4	(N/A) Ad-Rec	16	1-16	15-15	82.57M
RM4	Sparse Emb. Dim	32	1-32	15-15	165.10M
RM4	Top MLP	16	1-16	32-15-15	82.55M
RM4	Bottom MLP	16	1-8-16	15-15	82.55M
RM4	All Comp.	32	1-16-32	32-15-15	165.10M

D.8 Masking Analysis

We randomly sampled test samples from each dataset to examine the impact of masking on each attention head and its role in enhancing predictions by eliminating irrelevant features. We then plotted the attention weights for each attention head with and without a mask. Figures 14,15, and16 showcase the attention weights for the RM1, RM2, and RM3 models, respectively. These plots provide insights into how different attention heads learn feature interactions in multiple subspaces and how attention weights vary across different attention heads. By employing different mask values for each attention head, masking selectively masks out attention weights of irrelevant features, redistributing the attention weight across other relevant features accordingly.

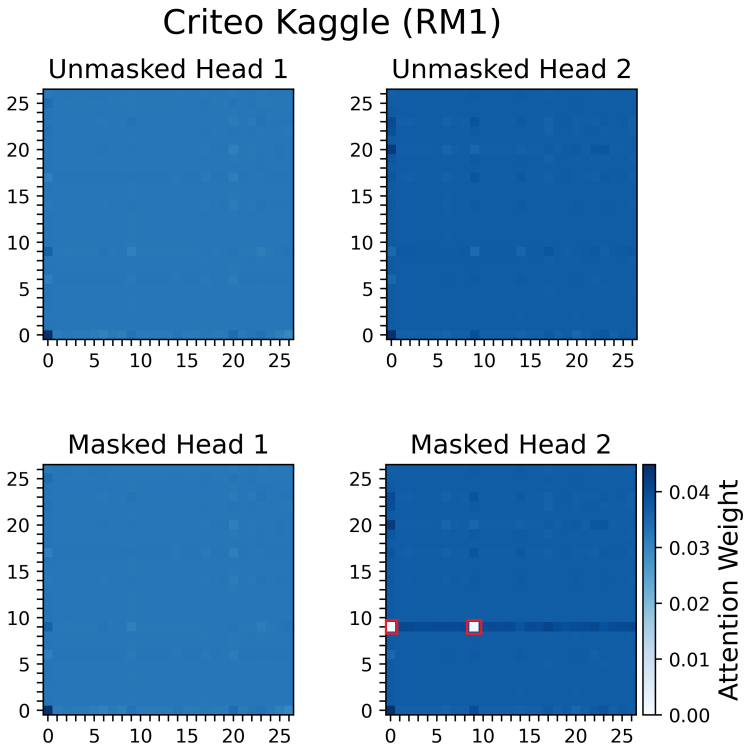


Figure 14: Attention Weights for **RM1** model across 2 heads. The unmasked head contains original attention weights, while the masked head contains attention weights after masking. X and Y axes contain 27 features with 0 as dense feature vectors, while others are sparse feature vectors. Highlighted features are the masked features that are irrelevant.

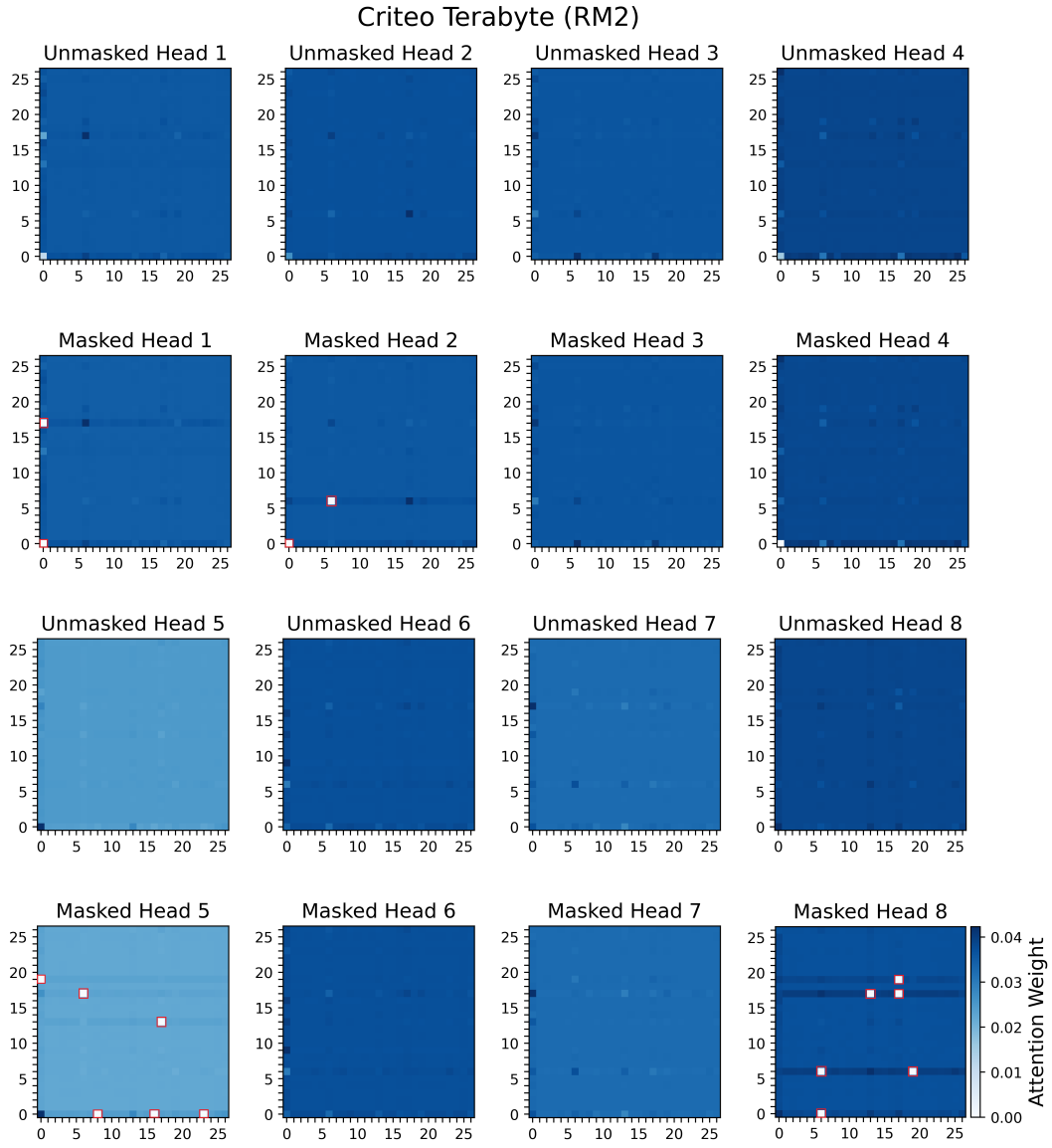


Figure 15: Attention Weights for **RM2** model across 8 heads. The unmasked head contains original attention weights, while the masked head contains attention weights after masking. X and Y axes contain 27 features with 0 as dense feature vectors, while others are sparse feature vectors. Highlighted features are the masked features that are irrelevant.

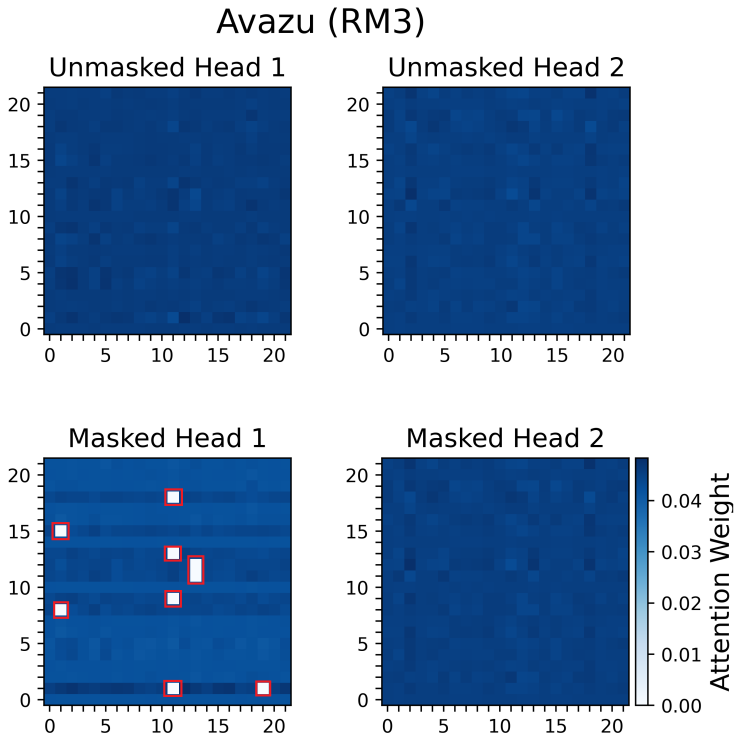


Figure 16: Attention Weights for **RM3** model across 2 heads. The unmasked head contains original attention weights, while the masked head contains attention weights after masking. X and Y axes contain 22 features with 0 as dense feature vectors, while others are sparse feature vectors. Highlighted features are the masked features that are irrelevant.