Deep Mutual Learning across Task Towers for Effective Multi-Task Recommender Learning

Yi Ren
Tencent
Beijing, China
yiren_bj@outlook.com

Ying Du
Tencent
Beijing, China
yingdu@tencent.com

Bin Wang
Tencent
Beijing, China
hillmwang@tencent.com

Shenzheng Zhang
Tencent
Beijing, China
qjzcyzhang@tencent.com

ABSTRACT

Recommender systems usually leverage multi-task learning methods to simultaneously optimize several objectives because of the multi-faceted user behavior data. The typical way of conducting multi-task learning(MTL) is to establish appropriate parameter sharing across multiple tasks at lower layers while reserving a separate task tower for each task at upper layers. With such design, the lower layers intend to explore the structure of task relationships and mine valuable information to be used by the task towers for accurate prediction.

Since the task towers exert direct impact on the prediction results, we argue that the architecture of standalone task towers is suboptimal for promoting positive knowledge sharing. First, for each task, attending to the input information of other task towers is beneficial. For instance, the information useful for predicting the "like" task is also valuable for the "buy" task. Furthermore, because different tasks are inter-related, the training labels of multiple tasks should obey a joint distribution. It is undesirable for the prediction results for these tasks to fall into the low density areas. Accordingly, we propose the framework of **D**eep **M**utual **L**earning across task towers(**DML**), which is compatible with various backbone multi-task networks. At the entry layer of the task towers, the shared component of Cross Task Feature Mining(*CTFM*) is introduced to transfer input information across the task towers while still ensuring one task's loss will not impact the inputs of other task towers. Moreover, for each task, dedicated network component called Global Knowledge Distillation(*GKD*) are utilized to distill valuable knowledge from the global results of the upper layer task towers to enhance the prediction consistency. Extensive offline experiments and online A/B tests are conducted to evaluate and verify the proposed approach's effectiveness.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Multi-task learning**.

KEYWORDS

Recommender Systems; Multi-Task Learning; Parameter Sharing

1 INTRODUCTION

Recently, we have seen the widespread application of recommender systems, which involve different types of user feedback signals, such as clicking, rating, commenting, etc. Moreover, no single feedback signal can accurately reflect user satisfaction. For example, over-concentrating on clicking may aggravate the click-bait issue. Therefore, it is highly desirable to be able to effectively learn and estimate multiple types of user behaviors at the same time. And Multi-Task Learning is a promising technique to address this challenge. Given several related learning tasks, the goal of multi-task learning is to enhance the overall performance of different tasks by leveraging knowledge transfer among tasks. With the Multi-Task Learning(MTL) paradigm[3, 25], multiple tasks are learned simultaneously in a single model. Compared with single-task solutions, MTL costs much fewer machine resources and improves learning efficiency since we just need to train and deploy a single model. Moreover, MTL usually can warrant enhanced recommendation performance through appropriate parameter sharing.

Most existing methods of Multi-Task Learning(MTL) appropriately share parameters across multiple tasks at lower layers while keeping separate task towers at upper layers. These methods can be roughly classified into four categories. The first category comprises the methods of **_hard parameter sharing_**. Among them, embedding sharing is the most intuitive structure to share information. For instance, the ESSM model [13] shares embedding parameters between the tasks of CTR (Click-Through Rate) and CVR (Conversion Rate) for improving the prediction performance of the sparse CVR task. In addition to the embedding parameters, the Shared-Bottom structure[3] is introduced to share the parameters of lower-layer MLPs among tasks. But these methods are severely plagued by the task conflicts and negative transfer issue. Second, for the methods of **_soft parameter sharing_**, each task owns separate parameters, which are regularized during training to minimize the differences between the shared parameters. L2-constrained[6] is a typical algorithm belonging to this category. Third, for the methods of **_customized routing_**, they learn customized routing weights for each task to combine and fuse information from lower-layer networks to counteract the negative transfer issue. Cross-stitch network[15] and sluice network[16] learn separate linear weights for each task to selectively merge representations from different

lower-level branches. SNR[11] modularizes the shared low-level layers into parallel sub-networks and uses a transformation matrix multiplied by a scalar coding variable to learn their connections to upper-level layers to alleviate the task conflict and negative transfer issue. MSSM[5] learns differing combinations of feature fields for each expert and designs finer-grained sharing patterns among tasks through a set of coding variables that selectively choose which cells to route for a given task. But the learned routing parameters of these methods are static for all the samples, which can hardly warrant optimal performance. Finally, the methods of **dynamic gating** learn optimized weights for each task based on the input sample to effectively combine the outputs of lower-level networks and achieve success in industrial applications. The MMoE [12] model adapts the Mixture-of-Experts (MoE)[9] structure to multi-task learning by sharing the expert sub-networks across all tasks, while also maintaining separate gating network optimized for each task. And Zhao et al. [26] extend the MMoE model [12] and apply it to learn multiple ranking objectives in Youtube video recommender systems. Moreover, PLE [17] achieves superior performance for news recommendation by assigning both shared expert sub-networks among tasks and dedicated expert sub-networks for each task.

AITM [24] is the most similar method, which also augments the arcitecture of task towers. Nevertheless, as a concrete implementation, it is not validated to enhance the performance of various multi-task models. Moreover, it can only work for the tasks with sequential dependence relations.

Admittedly, the aforementioned methods achieve impressive performance. However, as the task towers exert a direct effect on the prediction results, the standalone task towers tend not to be the most effective design for promoting positive knowledge transfer by exploiting the task relationships. First, for each task, the information selected by the relevant tasks is extremely valuable. Accordingly, we introduce the shared component of Cross Task Feature Mining(*CTFM*), which utilizes delicate attention mechanisms to extract relevant information from other tasks at the entry layer of the task tower. With the common attention mechanisms, the explicit task-specific information distilled by lower-level networks are mingled together and one task's loss will undesirably affect the inputs of other task towers, which is the task awareness missing problem and can hinder the learning of lower-level networks. In contrast to the usual attention mechanisms, our design can ensure appropriate information separation. We argue that reserving explicit task-specific knowledge has a positive effect on performance, which is validated in the experimental section. Second, because the tasks for recommender systems are related, the training labels of multiple tasks should obey a joint distribution. The prediction results for these tasks should not densely fall into the low-density areas. Therefore, a dedicated network named Global Knowledge Distillation(*GKD*) is introduced for each task to distill valuable global knowledge from the results of the upper layer task towers. For each task, the distilled global information helps to ensure consistent predictions with other tasks. We summarize our main contributions below.

- We propose the framework of **D**eep **M**utual **L**earning across task towers(**DML**), which is compatible with various backbone multi-task models.
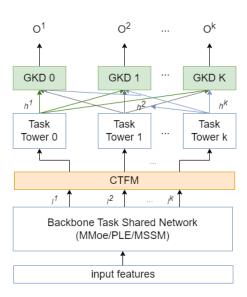

Figure 1: Overall Model Structure of DML

- The proposed novel sharing structure helps to enhance effective knowledge transfer across different tasks.
- We conduct offline experiments and online A/B testing to evaluate and understand the effectiveness of our method.

2 METHODOLOGY

In this section, we first introduce the problem of multi-objective ranking for recommender systems. Second, we describe the general design of DML. Finally, we elaborate on the introduced components.

2.1 Multi-Objective Ranking for Recsys

Given a set of candidates with N items $C = \{i_n\}_{1 \le n \le N}$, the ranking model for recommender systems is to rank and recommend the top M items $S = \{i_m\}_{1 \le m \le M} \subseteq C$ for user u so as to optimize the overall utility and enhance user experience. First, for each pair of user u and item i_n, the input feature x_n is derived. Second, a multi-task learning model is utilized to estimate K objectives corresponding to multiple user feedback signals. Furthermore, to compute the overall reward, we need to merge the multiple predictions with a function Φ shown in equation (2) to derive the item's final reward score for greedy ranking.

$$o_n^1, o_n^2, ..., o_n^K = MTLModel(x_n; \theta) \tag{1}$$

$$r_n = \Phi(o_n^1, o_n^2, ..., o_n^K) \tag{2}$$

where θ denotes the model parameters and Φ is usually a function manually tuned to reflect the reward of recommending item i_n to user u based on the business goals. With the estimated reward scores $\{r_n\}_{1 \le n \le N}$ for the items in C, the ranking model can recommend the item sequence S consisting of the top M items to the user u.

2.2 Overall Design of DML

Please refer to figure 1 for the overall network architecture of DML. With the existing MTL algorithms[25], the equation (1) can be

further decomposed as below. For simplicity, we omit the subscript n in this section.

$$l^1, l^2, ..., l^K = G(x; \theta_l) \qquad (3)$$

$$o^k = F^k(l^k; \theta_h^k) \qquad (4)$$

where G represents the lower level networks that encodes x to K different latent spaces with partially or fully shared parameters θ_l. And F^k is the upper-level network for task k, which accepts l^k as input to model objective k with task-specific higher level parameter θ_h^k. Multiple candidate models [5, 6, 11–13, 15–17, 25, 26] are proposed to enhance G with different parameter sharing designs.

In this research, rather than G, we focus on the enhancement of upper-level networks for improved prediction performance. First, the shared component of *CTFM* is introduced, which leverages the attention mechanism to extract relevant information from the inputs of other task towers (the results of Equation (3)) as a complement to the target task. Please note that this attention is well-designed to solve the task-awareness missing issue, for which the excessive encouragement of knowledge sharing is not conducive to the extraction of task-specific knowledge. With our design, the gradients computed from the target task's loss will not impact the inputs of other task towers.

$$\hat{l}^1, \hat{l}^2, ..., \hat{l}^K = CTFM(l^1, l^2, ..., l^K; \theta_s) \qquad (5)$$

where shared parameters θ_s is employed across different tasks.

Moreover, a separate multi-layer network is introduced for each task to process each element of $\{\hat{l}^k\}_{1 \le k \le K}$ and generate the hidden representation, based on which accurate prediction can be made.

$$h^k = H^k(\hat{l}^k; \theta_{t_0}^k) \qquad (6)$$

where H^k denotes the task-specific MLP for task k with separate parameters $\theta_{t_0}^k$.
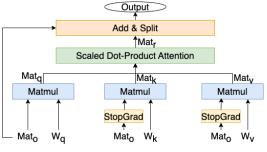
Finally, for each task, a dedicated component named *GKD* is utilized to distill information from the hidden representations for both itself and other tasks to promote prediction consistency across tasks and more precisely model the corresponding objective.

$$o^k = GKD^k(h^k, \{h^j\}_{1 \le j \le K}; \theta_{t_1}^k) \qquad (7)$$

where GKD^k is the dedicated component for task k. In contrast to *CTFM* at Equation (5), we utilize task-specific parameters $\theta_{t_1}^k$ here as specialization is usually helpful for upper layer networks. Furthermore, proper operation is implemented to ensure the prediction error of one task does not impact the hidden representations of other tasks.

2.3 Cross Task Feature Mining

Please refer to Figure 2(a) for the detailed process of *CTFM*. From the perspective of task towers, the outputs from lower-level networks at Equation (3) can be recognized as the mined features for them. First, for each task, the features mined for related tasks can be leveraged to enhance its prediction performance. Thus, trainable task embeddings, namely $\{t^k\}_{1 \le k \le K}$, are derived for the tasks to facilitate the learning of general task relations. Second, the importance of features from related tasks can vary per sample. Accordingly, we stack together the items of the set $\{l^k + t^k\}_{1 \le k \le K}$ to derive the matrix $Mat_o \in \mathcal{R}^{K \times d_0}$ where d_0 is the size of l^k and t^k. Third, we use

(a) Cross Task Feature Mining (Shared)

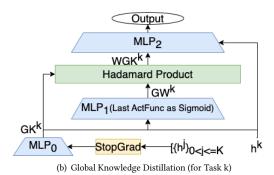(b) Global Knowledge Distillation (for Task k)

Figure 2: Introduced Components of DML

the projection parameters $W_q, W_k, W_v \in \mathcal{R}^{d_0 \times d_0}$ to transform Mat_o to the query, key, and value matrix of $Mat_q, Mat_k, Mat_v \in \mathcal{R}^{K \times d_0}$. And gradient backpropagation from Mat_k and Mat_v to Mat_o is forbidden. Moreover, the scaled dot-product attention [18] is performed on Mat_q, Mat_k, and Mat_v to compute the result matrix $Mat_r \in \mathcal{R}^{K \times d_0}$. Finally, we add Mat_o to Mat_r for residual connection and further split based on the first axis to return $\{\hat{l}^k\}_{1 \le k \le K}$. Please note that the aforementioned networks are shared among the tasks to encourage generalizable modeling with parameter sharing. The usual attention mechanisms will cause the task awareness missing problem and can hinder the learning of lower-level networks. Instead, our design can ensure appropriate information separation and reserve the explicit task-specific knowledge by stopping the gradient backflow from Mat_k and Mat_v to Mat_o.

2.4 Global Knowledge Distillation

Please refer to Figure 2(b) for the detailed process of *GKD*. In contrast to *CTFM*, each task is assigned dedicated parameters for global knowledge distillation. Acting as the last step, we would like to facilitate more flexible modeling by promoting task specialization here. This module accepts the hidden representations for both the corresponding task and other tasks as input. First, a multilayer perceptron(MLP) is utilized to extract valuable global knowledge ($GK^k \in \mathcal{R}^{d_1}$) for the target task k from the concatenation of all these hidden representations ($\{h^j\}_{1 \le j \le K}$). Since the design goal here is to train task-specific MLPs to distill relevant global knowledge while not impacting the hidden representations, the gradient

Table 1: The overall performance. The bold-face font denotes the winner in that column. Moreover, the "*" symbol denotes introducing DML achieves significant (p < 0.05 for one-tailed t-test) gain over the corresponding baseline.

Model	ML-1M			Electronics	
	AUC	MSE	Consistent Ratio	AUC_{rate}	AUC_{pos}
Single Task	0.8066	0.7741	0.7154	0.7608	0.7334
SB	0.8100	0.7724	0.7530	0.7876	0.7608
SB+DML	0.8115*	0.7648*	**0.7649***	0.7890*	0.7631*
MSSM	0.8128	0.7651	0.7519	0.7883	0.7627
MSSM+DML	0.8141*	0.7611*	0.7637*	0.7892*	0.7641*
MMOE	0.8105	0.7688	0.7507	0.7888	0.7628
MMOE+DML	0.8135*	0.7591*	0.7596*	**0.7897***	**0.7644***
PLE	0.8122	0.7606	0.7514	0.7885	0.7627
PLE+DML	**0.8151***	**0.7533***	0.7631*	0.7893*	0.7640*

Table 2: Further Analysis Results

Model	ML-1M		Electronics	
	AUC	MSE	AUC_{rate}	AUC_{pos}
$MSSM$	0.8128	0.7651	0.7883	0.7627
$MSSM + CTFM$	0.8134	0.7654	0.7891	0.7636
$MSSM + GKD$	0.8129	0.7636	0.7886	0.7633
$MSSM + DML_{v0}$	0.8138	0.7625	0.7884	0.7631
$MSSM + DML$	**0.8141**	**0.7611**	**0.7892**	**0.7641**
PLE	0.8122	0.7606	0.7885	0.7627
$PLE + CTFM$	0.8133	0.7603	0.7890	0.7633
$PLE + GKD$	0.8137	0.7574	0.7887	0.7634
$PLE + DML_{v0}$	0.8141	0.7536	0.7886	0.7629
$PLE + DML$	**0.8151**	**0.7533**	**0.7893**	**0.7640**

backpropagation from GK^k to the MLP's input is prohibited. Second, we input GK^k and h^k to another MLP with Sigmoid as its last activation function. Then, the weights of GK^k's different latent dimensions can be dynamically adapted for each sample. The weights are denoted by $GW^k \in \mathcal{R}^{d_1}$. Moreover, the weighted global knowledge ($WGK^k \in \mathcal{R}^{d_1}$) is computed with the hadamard product of GK^k and GW^k. Finally, WGK^k and h^k are concatenated together as the input for the last MLP to make predictions for task k.

3 EXPERIMENTS

In this section, we conduct extensive offline experiments[1] and online A/B testing to prove DML's effectiveness.

3.1 Experimental Settings for Public Data

3.1.1 Datasets. We evaluate our methods on two public datasets.

- **MovieLens-1M**[8]: One of the currently released MovieLens datasets, which contains 1 million movie ratings from 6,040 users on 3,416 movies.
- **Amazon**[14]: A series of datasets consisting of product reviews from Amazon.com. We use the sub-category of "Electronics" including 1.7 million reviews from 192,403 users on 63,001 items.

For ML-1M, we introduce the binary classification task of positive rating prediction (>=4) and the regression task of rating estimation. These two tasks are strictly correlated. For electronics, following [20], we first augment the dataset by randomly sampling un-rated items for every user. Moreover, we make sure the number of the un-rated items is the same as the number of the rated items for each user. Furthermore, we introduce two binary classification tasks, namely rating prediction (whether a rating exists) and positive rating prediction. Compared with the tasks of ML-1M, the negative transfer is more likely to occur as the task relationship here is more complex(The pearson correlation coefficient [21] of these two labels is around 0.7). Both data are randomly split into the training set, validation set, and test set by the ratio of 8:1:1.

---

[1]The code can be found at: https://github.com/renyi533/mtl-consistency/tree/main.

3.1.2 Evaluation Metrics. The merge function Φ in Equation (2) assumes that the model can estimate accurate interaction probabilities for binary classification tasks (e.g. clicking) and absolute values for regression tasks (e.g. watch time). Therefore, instead of the ranking metrics, such as NDCG [19] and MRR [22], we use the metrics of AUC [7] for classification tasks and Mean Squared Error (MSE) [23] for regression tasks. Please note that many other recommendation literature, such as [5, 17, 26], also use similar metrics. For AUC, a bigger value indicates better performance. While, for MSE, it is the smaller the better.

3.1.3 Models. As **soft parameter sharing** methods need resources to store and train multiple sets of parameters, they are not widely applied in recommender systems. Thus, we select base models to cover the other three categories. The models include Shared-Bottom(SB)[3], MSSM[5], MMOE[12], and PLE[17]. MSSM is a recent method belonging to the **Customized Routing** category and achieves better results than SNR[11] and Cross-Stitch[15]. Though with the same category of **Dynamic Gating**, both MMOE and PLE are tested owing to their popularity. For each base model, we will verify whether DML can achieve additional gains. For reference, we also provide the performance of single task models.

3.1.4 Implementation Details. For each feature, we use the embedding size 8. As suggested by the original papers, we use 1 level bottom sub-networks for MMOE, MSSM, and SB while 2 levels for PLE. For SB, a sub-network of 1 layer structure with 128 output dimensions is shared by the tasks. For other multi-task models, each bottom level includes three sub-networks, which have the same aforementioned architecture. For MSSM and PLE, task-specific and shared sub-networks are designated. For multi-task models, each task tower is of the three layers MLP structure (128,80,1) and each task is assigned equal loss weight. For the single-task model, each task utilizes the four layers MLP structure (128,128,80,1). For the first two MLPs at Figure 2(b), we utilize the one layer structure with 80 as the output dimension. For the last MLP at Figure 2(b), a one layer structure with 1 as the output size is used. If not explicitly specified, RELU [2] is used as the default activation function. All models are implemented with tensorflow [1] and optimized using the Adam [10] optimizer with learning rate 0.001 and mini-batch size 512. We run 20 times for each test to report the results.

3.2 Overall Performance for Public Data

Please refer to Table 1 for the overall results. First, DML achieves significant gains across all these tested multi-task models on the two public datasets, which shows DML's wide compatibility. Second, DML-enhanced PLE and MMOE get the best performance for MovieLens and Electronics respectively. Considering their wide application in recommender systems, the results are as expected. Third, the multi-task models perform better than the single task models thanks to the knowledge transfer between tasks.

Besides AUC and MSE, DML should help to foster task consistency with *CTFM* and *GKD*. As the tasks of MovieLens are rigorously correlated, we verify whether DML really enhances task consistency on this data. First, we construct pairs of samples with different rating scores and count the pair numbers. Second, we count the number of pairs, for which the prediction scores of both tasks are in the same pair order as the rating score. The enhancement of the pair order consistency among the two prediction scores and rating score should positively contribute to the performance. Then, we can compute the metric of 'Consistency Ratio'. The listed data in Table 1 agree with our anticipation. (For Shared-Bottom, we also observe more pairs, for which predictions of both tasks are in rating score's reverse order. This can explain its worse performance in spite of the better consistency ratio.)

3.3 Further Analysis on Public Data

We select the two latest algorithms of PLE[17] and MSSM[5] to appraise the value of *DML*'s components, namely *CTFM* and *GKD*. Without the stop gradient operation, *CTFM* will be very similar to the common attention mechanism. To prove the benefit of *CTFM*'s design, we also add the assessment for DML_{v0}, which reserve the design of *GKD* while remove the gradient blocking operation of *CTFM*. Please refer to Table 2 for the evaluation results. First, *CTFM* and *GKD* both contribute considerable gains over the base model. Second, as the integrated model, *DML* enhances the performance further. Third, DML_{v0} is consistently worse than *DML*, which corroborates the value of reserving task-awareness. Compared with *CTFM* and *GKD*, DML_{v0} performs better on MovieLens while much worse on Electronics. The task relationship of Electronics is more complex and negative transfer across tasks usually exhibits more severe impact due to task conflicts. In this case, compared with vanilla attention, *CTFM* obtains substantial gains.

3.4 Online A/B Testing

DML is applied to the ranking stage[4] of an industrial large-scale news recommender system. PLE [17] is utilized as the base model. And the main prediction tasks are the binary classification task of Click Through Rate (CTR) and the regression task of item watch time. First, after the model converge by training with billions of samples, the AUC metric for CTR consistently increases 0.12% and the MSE metric for watch time decreases 0.14%. Moreover, the most important online metrics include effective PV(count of Page Views with watch time exceeding a threshold) and total watch time. We randomly distributed online users to two buckets with the base PLE model or PLE+DML model and evaluated the performance for two weeks. DML achieves significant (p<0.05) gains over the base model

by 1.22% for effective PV and 0.61% for total watch time. DML has been deployed to our online environment based on the results.

4 CONCLUSION

In this papaer, we propose the framework of **D**eep **M**utual **L**earning across task towers(**DML**), which is compatible with various backbone multi-task networks. Extensive offline experiments help to verify DML's effectiveness on multiple real-world datasets and across various base models. Moreover, thorough ablation studies are carried out to verify and understand the value of each newly introduced module. Finally, DML achieves significant online gains and has already been deployed to the online platform.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *Osdi*, Vol. 16. Savannah, GA, USA, 265–283.

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2018. Understanding Deep Neural Networks with Rectified Linear Units. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=B1J_rgWRW

[3] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[5] Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, et al. 2021. MSSM: a multiple-level sparse sharing model for efficient multi-task learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2237–2241.

[6] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*. 845–850.

[7] Peter A Flach, José Hernández-Orallo, and Cèsar Ferri Ramirez. 2011. A coherent interpretation of AUC as a measure of aggregated classification performance. In *ICML*.

[8] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[9] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[10] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[11] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. 2019. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 216–223.

[12] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[13] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. 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*. 1137–1140.

[14] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[15] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3994–4003.

[16] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4822–4829.

[17] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized

recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

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

[19] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, Vol. 8. 6.

[20] Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, Li Wei, and Ed H Chi. 2022. Can Small Heads Help? Understanding and Improving Multi-Task Generalization. In *Proceedings of the ACM Web Conference 2022*. 3009–3019.

[21] Wikipedia. 2023. Pearson correlation coefficient — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Pearson%20correlation%20coefficient&oldid=1146097966. [Online; accessed 15-April-2023].

[22] Wikipedia contributors. 2022. Evaluation measures (information retrieval) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Evaluation_measures_(information_retrieval)&oldid=1095286224. [Online; accessed 9-January-2023].

[23] Wikipedia contributors. 2022. Mean squared error — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=1127519968. [Online; accessed 31-January-2023].

[24] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3745–3755.

[25] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[26] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.