MSSM: A Multiple-level Sparse Sharing Model for Efficient Multi-Task Learning

Ke Ding*
Xin Dong*
Ant Group
Shanghai, China
{dingke.dk,zhaoxin.dx}@antgroup.com

Yong He
Lei Cheng†
Ant Group
Hangzhou, China
{heyong.h,lei.chenglei}@antgroup.com

Chilin Fu†
Zhaoxin Huan†
Ant Group
Hangzhou, China
{chilin.fcl,zhaoxin.hzx}@antgroup.com

Hai Li†
Tan Yan†
Ant Group
Shanghai, China
{tianshu.lh,yantan.yt}@antgroup.com

Liang Zhang†
Xiaolu Zhang†
Ant Group
Hangzhou, China
{zhuyue.zl,yueyin.zxl}@antgroup.com

Linjian Mo
Ant Group
Shanghai, China
linyi01@antgroup.com

ABSTRACT

Multi-task learning(MTL) is an open and challenging problem in various real-world applications. The typical way of conducting multi-task learning is establishing some global parameter sharing mechanism across all tasks or assigning each task an individual set of parameters with cross-connections between tasks. However, for most existing approaches, all tasks just thoroughly or proportionally share all the features without distinguishing the helpfulness of them. By that, some tasks would be intervened by the unhelpful features that are useful for other tasks, leading to undesired negative transfer between tasks. In this paper, we design a novel architecture named the Multiple-level Sparse Sharing Model (*MSSM*), which can learn features selectively and share knowledge across all tasks efficiently. *MSSM* first employs a field-level sparse connection module (*FSCM*) to enable much more expressive combinations of feature fields to be learned for generalization across tasks while still allowing for task-specific features to be customized for each task. Furthermore, a cell-level sparse sharing module (*CSSM*) can recognize the sharing pattern through a set of coding variables that selectively choose which cells to route for a given task.Extensive experimental results on several real-world datasets show that *MSSM* outperforms SOTA models significantly in terms of *AUC* and *LogLoss* metrics.

CCS CONCEPTS

• **Computing methodologies** → **Multi-task learning**; **Transfer learning**.

*Both authors contributed equally to this research.
†Authors are ordered by last name.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463022

KEYWORDS

Multi-task Learning; sparse connection; parameter sharing

ACM Reference Format:
Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, and Linjian Mo. 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 (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463022

1 INTRODUCTION

Multi-task learning (MTL) aims to solve multiple related tasks simultaneously and has achieved rapid growth in recent years. Compared to single-task learning, it can significantly improve learning efficiency and prediction accuracy by using shared representations to learn the common features between a collection of related tasks [1]. This allowed deployments of MTL in a wide range of real-world applications, including computer vision [3, 8], natural language processing [4, 9], online recommendation and advertising systems [12, 14, 18, 19].

A common modeling approach for multi-task learning focuses on establishing some sort of parameter sharing mechanism by sharing the whole network across all tasks, such as cross-stitch network [13] and sluice network [16]. More recently, another new trend for addressing multi-task learning is to assign each task an individual set of parameters with cross-talk connections between tasks, such as MMOE [12], SNR [11] and PLE [18]. The benefits of these shared architectures are multi-fold. Firstly, they exploit task relatedness with inductive bias learning. Secondly, by forcing tasks to share model capacity, they introduce a regularization effect and improve generalization. Last but not least, they offer much more compact and efficient model architectures compared with training each task separately.

However, such shared model architectures introduce new challenges for multi-task learning. Firstly, these models force all tasks to share the same hidden space, which limits its expressivity. Thus, especially for the input features, unhelpful or harmful features may

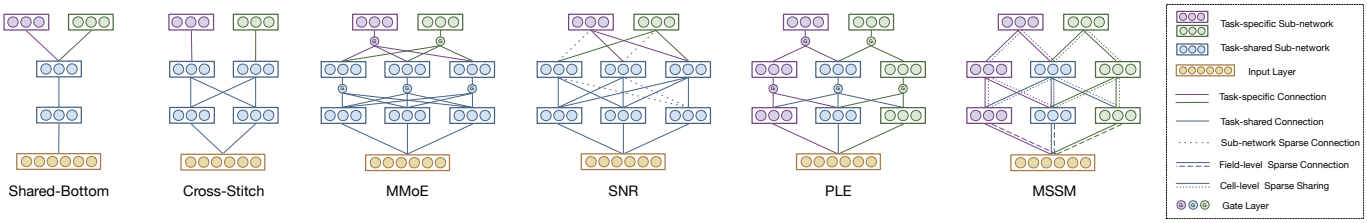


Figure 1: The overview of several SOTA MTL models including shared-bottom network [1], cross-stitch network [13], MMOE [12], SNR [11], PLE [18] and our MSSM model. Compared to SNR, the proposed MSSM adopts a multiple-level (including field-level and cell-level) sparse connections (solid line + dashed line) instead of sub-network connections.

be transported between tasks with the same importance as helpful ones, namely, interference is generated. Secondly, as the size of the network parameters grows proportionally with respect to the total number of tasks, these models are not computationally or memory efficient.

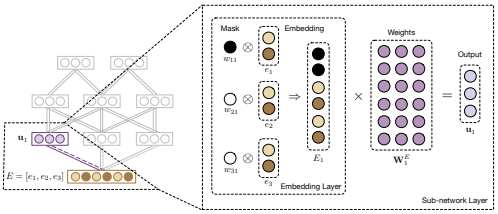
In this paper, to address the challenges above, we propose a novel and non-trivial architecture called a Multiple-level Sparse Sharing Model (*MSSM*), which can learn features selectively and share knowledge across all tasks in a flexible and efficient way. *MSSM* consists of two main components: a field-level sparse connection module (*FSCM*) and a cell-level sparse sharing module (*CSSM*). The *FSCM* employs a sparse mask to automatically determine the importance of the feature fields for the respective task and allow learning both task-shared and task-specific feature fields in an end-to-end manner. It enables much more expressive combinations of feature fields to be learned for generalization across tasks while still allowing for task-specific features to be customized for each individual task. The *CSSM* implements an efficient sharing architecture by employing a more fine-grained and cell-level sparse connection among sub-networks. This mechanism can learn the sharing pattern through some coding variable vectors that selectively choose which cells to route for a given task in the multi-task network. At last, we demonstrate that this architecture outperforms the state-of-the-art multi-task learning tasks in our experiments.

Specifically, the main contributions of this paper can be summarized as the following three aspects:

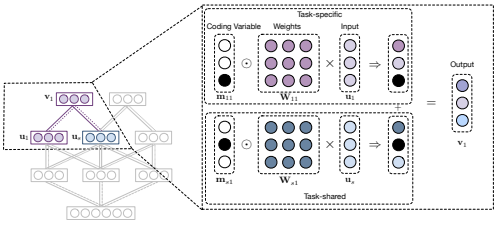
- We propose a novel and state-of-the-art architecture named the Multiple-level Sparse Sharing Model (*MSSM*), which can learn features selectively and share knowledge across all tasks in a flexible and efficient way.
- We implement a fine-grained and flexible parameter sharing policy, which uses a transformation matrix multiplied by a binary coding variable vector to control the connection among cells in the sub-networks of different layers.
- We conduct extensive experiments on several real-world datasets to evaluate the effectiveness of our proposed model. Experimental results show that our proposed model outperforms state-of-art methods in terms of *AUC* and *LogLoss* metrics.

2 METHODOLOGY

Given a set of K tasks, our goal is to seek a feature sharing mechanism that decides which features should be shared across tasks and which parts should be task-specific to improve the accuracy. Figure 1 illustrates an overview of our architecture with other multi-task learning frameworks. There are two types of sparse connections in *MSSM*, i.e., field-level and cell-level sparse connections. The details of these connections are presented in Section 2.1 and Section 2.2 respectively.



(a) Field-level Sparse Connection Module (FSCM)



(b) Cell-level Sparse Sharing Module (CSSM)

Figure 2: Visualization of FSCM and CSSM.

2.1 Field-level Sparse Connection Module

The Field-level Sparse Connection Module (*FSCM*) is designed to allow the network to learn task-related feature fields by applying a feature-field sparse mask to the input layer. As shown in Figures 1 and 2a, the *FSCM* in our architecture takes in the input layer. Consequently, the outputs of *FSCM* are fed into the *CSSM*. For the input layer, each feature field is embedding-transformed into low dimensional dense representations. We denote the representation of the ith feature field as e_i for all samples and the learned mask for task j as w_{ij}. The task-shared or task-specific features E_j are then computed by field-level multiplication of the masks with the input embedding-transformed features:

$$E_j = \prod_{i=1}^{n} w_{ij} \cdot e_i = [w_{1j} \cdot e_1, \cdots, w_{ij} \cdot e_i, \cdots, w_{nj} \cdot e_n],$$

where n is the dimension of the input embedding-transformed feature field and [denotes vector concatenation operation.

The sparse mask w_{ij}, indicating the importance of ith feature field for the respective task j, can be calculated as follows:

$$w_{ij} = relu(tanh(g_{ij} \cdot e_i)),$$

where g_{ij} is a learnable weight. Figure 2a presents the detail of the FSCM mechanism, where $\mathbf{W}_1^E$, $\mathbf{u}_1$ are the weight and output of the first-level sub-network, respectively. The mask, following a *tanh* and *relu* activation functions, can be learned in a self-supervised and end-to-end fashion with back-propagation. It masks zeros value of the input feature fields if $tanh(g_{ij} \cdot e_i) \leq 0$. Otherwise, the importance scores of non-zeros feature fields are assigned. Compared with the conventional MTL method which takes all the feature fields as input for each specific task equally, we propose a field-level sparse mask to automatically determine the importance of the feature fields for the respective task and allow learning both task-shared and task-specific feature fields in an end-to-end manner. And we show better results demonstrating this in Section 3.5.

2.2 Cell-level Sparse Sharing Module

In this section, we focus on improving MTL performance by a fine-grained and flexible parameter sharing policy. Several previous works [2, 6, 11–13, 17] addresses this goal by introducing a set of sub-networks and allowed sparse connections among these sub-networks, which improves the serving computational efficiency. However, all these works are subnetwork-level sparse connections, and all tasks share the whole sub-network. Accordingly, we draw lessons from these previous works and further upgrade a more fine-grained parameter sharing module in the multi-task model by controlling the connection among the more fine-grained cells in sub-networks of different layers. We call this architecture Cell-level Sparse Sharing Module (*CSSM*).

The key idea behind *CSSM* is to compute a Hadamard-product of a transformation matrix with a coding variable vector. We present the process of this Hadamard-product in Figure 2b. Suppose there are two subsequent layers of sub-networks, and a lower-level layer has 3 sub-networks, and the higher-level layer has 3 sub-networks. The outputs of the lower-level sub-networks are denoted as $[\mathbf{u}_1, \mathbf{u}_2]$ (task-specific) and $\mathbf{u}_s$ (task-shared). Consequently, the corresponding outputs of higher-level 3 sub-networks are $[\mathbf{v}_1, \mathbf{v}_2]$ (task-specific) and $\mathbf{v}_s$ (task-shared). Then the 3 higher-level sub-networks can be formulated as:

$$\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_s\end{bmatrix} = \begin{bmatrix}\mathbf{m}_{11} \odot \mathbf{W}_{11} & \mathbf{m}_{12} \odot \mathbf{W}_{12} & \mathbf{m}_{1s} \odot \mathbf{W}_{1s}\\ \mathbf{m}_{21} \odot \mathbf{W}_{21} & \mathbf{m}_{21} \odot \mathbf{W}_{22} & \mathbf{m}_{2s} \odot \mathbf{W}_{2s}\\ \mathbf{m}_{s1} \odot \mathbf{W}_{s1} & \mathbf{m}_{s2} \odot \mathbf{W}_{s2} & \mathbf{m}_{ss} \odot \mathbf{W}_{ss}\end{bmatrix}\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \mathbf{u}_s\end{bmatrix},$$

where $\mathbf{W}_{ij}$ is a transformation matrix from the jth lower-level sub-network to the ith higher-level sub-network, $\mathbf{m}_{ij}$ represents a coding variable vector (a group of binary variables controlling the cell-level sparse connection), and $\odot$ denotes Hadamard-product. Nevertheless, the values of $\mathbf{m}_{12}$ and $\mathbf{m}_{21}$ are set to 0 which mean there are no connections.

2.3 MSSM Architecture Learning

In practice, one key problem to learn our model is how to update each element m in coding variable $\mathbf{m}_{ij}$, where $m \in \{0, 1\}$. Similar to [11], we propose to model the coding variable vector $\mathbf{m}_{ij}$ as latent random variables from parameterized distributions and learn the distribution parameters with model parameters simultaneously. Therefore, each coding variable in $\mathbf{m}_{ij}$ is supposed to be drawn from a latent policy distribution, such as Bernoulli distributions. Then, the distribution parameters can be trained by gradient-based optimization together with model parameters using reparameterization trick [10, 15]. The code variables are used to control the connection of sub-networks with learnable latent variables. Accordingly, the process of learning *MSSM* is detailed in Algorithm 1, where bold indicates the variable vector.

Algorithm 1: Minimizing the joint loss with Adam.

Input :Hyper-parameter $\beta, \zeta, \gamma, \lambda_1, \lambda_2, \omega_1, \omega_2$ for two tasks;
Initialize Weights $\mathbf{W}_j^E$, $\mathbf{W}_{ij}$, $log(\boldsymbol{\alpha}_{ij})$, $\mathbf{G}_j = [g_{1j}, \ldots, g_{nj}]$;
for *each* $i \in [1, 2]$ *task* **do**
  $\mathbf{v}_i = 0$;
  for *each* $j \in [1, 2, s]$ *sub-network* **do**
    if $j \neq s$ ***and*** $i \neq j$ **then**
      continue;
    end
    Randomly sampling a group of values $\mathbf{u} \sim U(0, 1)$;
    $\mathbf{s}_{ij} = sigmoid((log(\mathbf{u}) - log(1 - \mathbf{u}) + log(\boldsymbol{\alpha}_{ij}))/\beta)$;
    $\mathbf{m}_{ij} = min(1, max(\mathbf{s}_{ij}(\zeta - \gamma) + \gamma, 0))$;
    $E_j = [relu(tanh(g_{1j} \cdot e_1)) \cdot e_1, \ldots, relu(tanh(g_{nj} \cdot e_n) \cdot e_n]$;
    $\mathbf{u}_j = \mathbf{W}_j^E \times E_j$;
    $\mathbf{v}_i = \mathbf{v}_i + \mathbf{m}_{ij} \odot \mathbf{W}_{ij} \times \mathbf{u}_j$;
  end
end
$L(\mathbf{X}, \mathbf{Y}_1, \mathbf{Y}_2) = \omega_1 L_1(\mathbf{v}_1, \mathbf{Y}_1) + \omega_2 L_2(\mathbf{v}_2, \mathbf{Y}_2) + \lambda_1 L_{reg_w} + \lambda_2 L_{reg_m}$;

2.4 Joint Loss Optimization for MTL

In general multi-task learning with K tasks, input $\mathbf{X}$ and task-specific labels $\mathbf{Y}_k$, $k = 1, 2, \cdots, K$. Therefore, we learn the model parameters by minimizing the aggregated loss as follows:

$$L(\mathbf{X}, \mathbf{Y}_{1:K}) = \sum_{k=1}^{K} \omega_k L_k(\mathbf{X}, \mathbf{Y}_k) + \lambda_1 L_{reg_w} + \lambda_2 L_{reg_m},$$

where L_k, ω_k are loss function and loss weight of task k respectively, L_{reg_w} is a L2 regularization term to prevent over-fitting, L_{reg_m} is a L0 regularization term on the latent variables to keep sparse connection like [10], λ_1, λ_2 are hyper-parameters.

3 EXPERIMENTS

In this section, we evaluate the performance of the proposed *MSSM* on two real-world datasets, and experimental comparison demonstrates the effectiveness of *MSSM* that outperforms state-of-art algorithms for multi-task learning.

3.1 Datasets

3.1.1 IJCAI CUP 2015 Dataset (Public). This public dataset comprises anonymized users' shopping logs in the past six months before and on the "Double 11" day. This dataset contains 219,973 users providing 1,862,193 samples about 219,182 items. Moreover, we construct two tasks for this dataset: Favorite and Purchase.

3.1.2 Alipay Advertising Dataset (Industrial). We collected user traffic logs from the Alipay advertising system.This dataset contains 1,838,449 users providing 2,672,481 samples about 3,731 advertisements. It also has two tasks: Following and Purchase.

Table 1: Performance comparison. * indicates the statistical significance for $p <= 0.05$ compared with the best baseline based on the paired t-test.

Model	IJCAI CUP 2015			Alipay Advertising		
	AUC1	AUC2	Logloss	AUC1	AUC2	Logloss
Single-Task	0.7245	0.7522	-	0.7309	0.7799	-
Shared-Bottom	0.7239	0.7518	0.6135	0.7275	0.7825	0.2906
Cross-Stitch	0.7242	0.7588	*0.6108*	0.7310	0.7782	0.2894
MMOE	0.7210	0.7508	0.6142	0.7317	0.7842	0.2893
SNR-Aver	*0.7253*	0.7522	0.6149	0.7295	0.7830	0.2897
SNR-Trans	0.7209	*0.7602*	0.6126	*0.7340*	0.7871	*0.2873*
PLE	0.7242	0.7536	0.6129	0.7319	*0.7875*	0.2890
MSSM	**0.7303***	**0.7704***	**0.6074***	**0.7386***	**0.7923***	**0.2859***

3.2 Baseline Models

We compare our *MSSM* with the following baseline models: (1) *Single-Task*: The model only uses a single task label. (2) *Shared-Bottom* [1]: This model shares several low-level network layers for all the tasks, and each task has its own task-specific high-level layers. (3) *Cross-Stitch* [13]: This model has low-level task-specific network layers and concatenated the low-level layers weighted by learnable parameters. (4) *MMOE* [12]: It splits the shared low-level layers into sub-networks and uses different gating networks for various tasks to utilize different sub-networks. (5) *SNR* [11]: It 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. (6) *PLE* [18]: It separates shared components and task-specific components and adopts a progressive routing mechanism to achieve more efficient information sharing.

3.3 Experimental Setup

3.3.1 Evaluation Metrics. For fair comparison, we employ the following evaluation metrics: (1) *AUC* [5]: It measures the goodness of order by ranking all the items with a predicted value in the test set. (2) *Logloss*: It is the total loss value of all K tasks on the test set.

3.3.2 Implementation Details. All the models are trained using the Adam optimizer [7] with an initial learning rate of 1e-4. The mini-batch size is fixed as 256. The hidden sizes of the two shared hidden layers in both single-task, and shared-bottom models are 96, 16, respectively. The number of sub-networks/experts in SNR, PLE and MMOE is set to 3, and the hidden size of each sub-network/expert is 32. For each specific task, there are two specific hidden layers with the size of 32, 16, respectively. The loss balance parameters ω_i, $i = 1, 2$, are set to 1, indicating training task losses are weighed equally. These regularization parameters λ_1 and λ_2 are set to 1e-3, 1e-2, respectively.

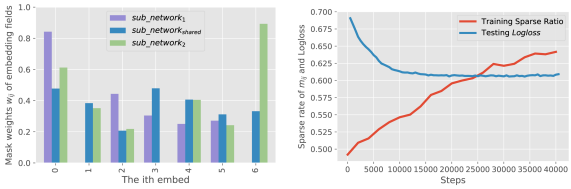
3.4 Experimental Results

Table 1 shows the *AUC* and *Logloss* results on the IJCAI CUP 2015 dataset and the Alipay Advertising dataset. We repeat all experiments 10 times and report the averaged results in the table.

We notice that negative transfer happens for Shared-Bottom, MMOE and SNR-Aver models, where the performance is worse than the single-task model. Making comparisons among all the MTL models, the MMOE model performs worse than SNR-Trans and PLE model. One possible reason may be that it is hard to decide which internal outputs of low-level layers should be used by the higher-level gating networks. Furthermore, we can see that our proposed *MSSM* achieves the highest *AUC* and lowest *Logloss* over all baseline models on all tasks, which validates the efficiency of our MSSM architecture.

Table 2: Performance comparison of the three variants.

Model	IJCAI CUP 2015			Alipay Advertising		
	AUC1	AUC2	Logloss	AUC1	AUC2	Logloss
MSSM-Basic	0.7209	0.7602	0.6126	0.7340	0.7871	0.2873
MSSM-a	0.7284	0.7643	0.6106	0.7376	0.7896	0.2867
MSSM-b	0.7285	0.7659	0.6099	0.7354	0.7899	0.2864
MSSM	**0.7303**	**0.7704**	**0.6074**	**0.7386**	**0.7923**	**0.2859**



(a) The mask weights of FSCM (b) The sparse ratio of CSSM

Figure 3: Visual Analysis on IJCAI CUP 2015 Dataset.

3.5 Empirical Analysis

3.5.1 Ablation Study. To verify the characteristics of the two levels of sparse connection modules of *MSSM*, accordingly, we come up with three variants of *MSSM* as follows: (1) *MSSM-Basic*. The *MSSM* model without *FSCM* and *CSSM*, which degenerates to SNR-Trans model. (2) *MSSM-a*. The *MSSM* model without *FSCM*. (3) *MSSM-b*. The *MSSM* model without *CSSM*.

The results of the ablation studies on all evaluation settings are reported in Table 2. We make the following observations: (1) *MSSM-a* and *MSSM-b* gains some improvements over *MSSM-basic* , which reveals the benefit of including field-level sparse connection of the feature fields and cell-level sparse parameter sharing module; (2) As the integrated model, *MSSM* improves the performance further by exploiting both the two levels of sparse connection modules.

3.5.2 Visual Analysis. Figure 3a shows values of all sparse mask weights w on IJCAI dataset. We can see that w_{11}, w_{32}, w_{61} are 0 which shows *FSCM* realizes sparse connection. Moreover, different mask weights indicate that *FSCM* can learn task-specific features for each task. What is more, as the model converges, the sparsity of the cell-level connections in *CSSM* is larger than 60% in Figure 3b.

4 CONCLUSION

In this paper, we present a novel and non-trivial architecture called the Multiple-level Sparse Sharing Model(*MSSM*), which can learn features selectively and share knowledge across all tasks in a flexible and efficient way. *MSSM* employs a field-level sparse connection module (*FSCM*) to automatically determine the importance of the feature fields for the respective task. Moreover, a cell-level sparse sharing module (*CSSM*) can learn the sharing pattern through a set of coding variables that choose which cells to execute for a given task in the multi-task network. We show that our proposed *MSSM* model outperforms all existing models for MTL tasks.

REFERENCES

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

[2] Xiaokai Chen, Xiaoguang Gu, and Libo Fu. 2020. Boosting share routing for multi-task learning. *arXiv preprint arXiv:2009.00387* (2020).

[3] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir A Rawashdeh. 2019. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.

[4] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. 160–167.

[5] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[6] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. 2020. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11543–11552.

[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1871–1880.

[9] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. (2015).

[10] Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning Sparse Neural Networks through L_0 Regularization. *arXiv preprint arXiv:1712.01312* (2017).

[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] 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.

[14] Zhen Qin, Yicheng Cheng, Zhe Zhao, Zhe Chen, Donald Metzler, and Jingzheng Qin. 2020. Multitask Mixture of Sequential Experts for User Activity Streams. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3083–3091.

[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[16] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *stat* 1050 (2017), 23.

[17] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2019. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423* (2019).

[18] 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.

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