MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks

Yun He
Texas A&M University
USA
yunhe@tamu.edu

Xue Feng
Meta AI
USA
xfeng@fb.com

Cheng Cheng
Meta AI
USA
cc6@fb.com

Geng Ji
Meta AI
USA
gji@fb.com

Yunsong Guo
Meta AI
USA
yunsong@fb.com

James Caverlee
Texas A&M University
USA
caverlee@tamu.edu

ABSTRACT

In many personalized recommendation scenarios, the generalization ability of a target task can be improved via learning with additional auxiliary tasks alongside this target task on a multi-task network. However, this method often suffers from a serious optimization imbalance problem. On the one hand, one or more auxiliary tasks might have a larger influence than the target task and even dominate the network weights, resulting in worse recommendation accuracy for the target task. On the other hand, the influence of one or more auxiliary tasks might be too weak to assist the target task. More challenging is that this imbalance dynamically changes throughout the training process and varies across the parts of the same network. We propose a new method: MetaBalance to balance auxiliary losses via directly manipulating their gradients w.r.t the shared parameters in the multi-task network. Specifically, in each training iteration and adaptively for each part of the network, the gradient of an auxiliary loss is carefully reduced or enlarged to have a closer magnitude to the gradient of the target loss, preventing auxiliary tasks from being so strong that dominate the target task or too weak to help the target task. Moreover, the proximity between the gradient magnitudes can be flexibly adjusted to adapt MetaBalance to different scenarios. The experiments show that our proposed method achieves a significant improvement of 8.34% in terms of NDCG@10 upon the strongest baseline on two real-world datasets. The code of our approach can be found at here.[1]

CCS CONCEPTS

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

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

*A majority of this work was done while the first author was interning at Meta AI.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512093

KEYWORDS

Multi-Task Learning, Auxiliary Learning, Personalized Recommendation, Gradient-based Optimization

ACM Reference Format:
Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. 2022. MetaBalance: Improving Multi-Task Recommendations via Adapting Gradient Magnitudes of Auxiliary Tasks. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512093

1 INTRODUCTION

The accuracy of personalized recommendations can often be improved by transfer learning from related auxiliary information. For example, a primary task on e-commerce platforms like Amazon and eBay is to predict if a user will purchase an item. This purchase prediction task can benefit from transferring knowledge about the user's preference from auxiliary information like which item URLs the user has clicked and which items the user has put into the shopping cart. A common way to enable such transfer learning is to formulate this auxiliary information as auxiliary tasks (e.g., predict if a user will click a URL) and optimize them jointly with the target task (e.g., purchase prediction) on a multi-task network. In this way, knowledge can be transferred from the auxiliary tasks to the target task via the shared bottom layer of the multi-task network as shown in Figure 1(a). Enhanced with auxiliary information, the target task can obtain better performance than training the target task in isolation. Since the motivation of introducing those auxiliary tasks is often to purely assist the target task, in this paper, we focus on scenarios where only the performance of the target task is of interest.

Beyond purchase prediction, many other recommendation scenarios [1, 3, 9, 11, 20–22, 33, 35] can also benefit from such transfer learning from auxiliary tasks. In social recommendation [9, 21, 33], knowledge can be transferred from the social network to improve personalized recommendations via training the target task simultaneously with auxiliary tasks like predicting the connections or trust among users. To better estimate post-click conversion rate (CVR) in online advertising, related information like post-view click-through rate (CTR) and post-view click-through & conversion rate (CTCVR) can be introduced as auxiliary tasks [22]. Another example is that learning user and item embeddings from product review text can be

designed as auxiliary tasks to improve the target goal of predicting ratings on e-commerce platforms [35].

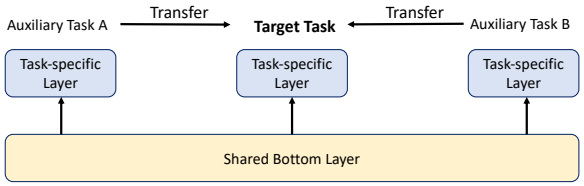
However, a key challenge to transfer learning from auxiliary tasks in personalized recommendation is the potential for a significant *imbalance of gradient magnitudes*, which can negatively affect the performance of the target task. As mentioned before, such transfer learning is often conducted on a multi-task network, which is commonly composed of a bottom layer with shared parameters and several task-specific layers. In training, each task has a corresponding loss and each loss has a corresponding gradient with respect to the shared parameters of that multi-task network. The sum of these gradients (for the target task and the auxiliary tasks) impacts how the shared parameters are updated. Hence, the larger the gradient is, the greater the impact this gradient has on the shared parameters. As a result, if the gradient of an auxiliary loss is much larger than the gradient of the target loss, the shared parameters will be most impacted by this auxiliary task rather than the target task. Consequently, the target task could be swamped by the auxiliary tasks, resulting in worse performance. On the other hand, if an auxiliary gradient is much smaller than the target gradient, the influence of this auxiliary task might be too weak to assist the target task. This *imbalance of gradient magnitudes* is common in industrial recommender systems: Figure 1(b) and 1(c) highlight two examples from Alibaba, which respectively demonstrate how the target task gradient can be dominated by an auxiliary task, and how some auxiliary tasks have gradients so small that they may only weakly inform the target task.

So how can we overcome this gradient imbalance? A simple and often used approach is to tune the weights of task losses (or gradients) through a grid or random search. However, such fixed task weights are not optimal because the gradient magnitudes change dynamically throughout the training and the imbalance might vary across the different subsets of the shared parameters as shown in Figure 1. Besides, it is time-consuming to tune the weights for multiple auxiliary tasks.

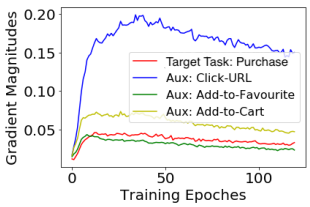
In this paper, we propose MetaBalance as a novel algorithm and flexible framework that adapts auxiliary tasks to better assist the target task from the perspective of gradient magnitudes. Specifically, MetaBalance has three strategies: (A) Strengthening the dominance of the target task – auxiliary gradients with larger magnitudes than the target gradient will be carefully reduced in each training iteration; (B) Enhancing the knowledge transferring from weak auxiliary tasks – auxiliary gradients with smaller magnitudes than the target gradient will be carefully enlarged; and (C) MetaBalance adopts both (A) and (B) in the same iteration. In the absence of sufficient prior knowledge, which strategy to apply is treated as a data-driven problem, where the best strategy can be empirically selected based on the performance over the validation set of the target task.

Moreover, MetaBalance has three key characteristics:

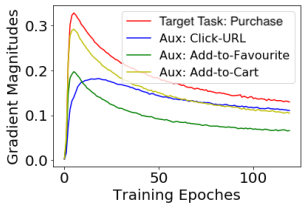
(1) Auxiliary gradients can be balanced *dynamically* throughout the training process and *adaptively* for different subsets of the shared parameters, which is more flexible than fixed weights for task losses;
(2) MetaBalance *prioritizes* the target task via preventing auxiliary tasks from being so strong that they dominate the target task or too weak to help the target task, which can be easily monitored by choosing one of the three strategies;
(3) The next important question is *how much should the auxiliary gradient magnitudes be reduced or enlarged?* We design a relax factor to control this to flexibly adapt MetaBalance to different scenarios. The relax factor can also be empirically selected based on the performance over the validation dataset of the target task.



(a) Transfer Learning from Auxiliary Tasks to Improve the Target Task on a Multi-task Network



(b) Imbalance on one part of the shared parameters (e.g., MLP layer)



(c) Imbalance on another part of the shared parameters (e.g., embedding layer)

Figure 1: The imbalance of gradient magnitudes in transfer learning from auxiliary tasks for recommendations on Alibaba data. The magnitudes dynamically change throughout the training, with the imbalance varying across different parts of the same multi-task network: in Fig 1(b), the gradient of auxiliary task click-URL is much larger than the target gradient; in Fig 1(c), the gradient of auxiliary task Add-to-Favorite is much smaller than the target gradient.

In sum, MetaBalance provides a flexible framework for adapting auxiliary gradients to better improve the target task from the perspective of gradient magnitudes. Extensive experiments over two real-world user behavior datasets from Alibaba show the effectiveness and flexibility of MetaBalance. In particular, we have four target observations:

- With the best strategy and relax factor selected from the validation set, MetaBalance can significantly boost the test accuracy of the target task, which shows that auxiliary knowledge can be better transferred to the target task via MetaBalance.
- MetaBalance can significantly outperform previous methods for adapting auxiliary tasks to improve the target task. For example, we observe a significant improvement of 8.34% upon the strongest baselines in terms of NDCG@10.
- Only one hyper-parameter in MetaBalance (the relax factor) needs to be tuned, irrespective of the number of tasks. Hence, MetaBalance requires only a few training runs, which is more efficient than tuning the weights of task losses, which can be computationally intensive as the number of tasks increases.

- MetaBalance can collaborate well with several popular optimizers including Adam, Adagrad and RMSProp, which shows the potential that MetaBalance can be widely applied in many scenarios.

2 RELATED WORK

Recommendations with Auxiliary Tasks. In many personalized recommendation scenarios, the test accuracy of the target task can be improved via joint learning with auxiliary tasks. In social recommendation [9, 21, 33], the knowledge about the user preference can be transferred from social network to the improve recommendations while the target task like rating prediction jointly train with auxiliary tasks like predicting the connections and trust among users. To improve post-click conversion rate (CVR) prediction, Ma et al. [22] consider the sequential pattern of user actions and introduce post-view click-through rate (CTR) and post-view click-through&conversion rate (CTCVR) as auxiliary tasks. To enhance music playlists or booklists recommendations, predicting if a user will like an individual song or book can also be used as auxiliary tasks and jointly learn with the list-based recommendations. Besides, Bansal et al. [1] design auxiliary tasks of predicting item meta-data (e.g., tags, genres) to improve the rating prediction as the target task. To improve the target goal of predicting ratings, learning user and item embeddings from product review text can also be designed as auxiliary tasks [35].

Auxiliary Learning. In this paper, we focus on transferring knowledge from auxiliary tasks to improve the target recommendation task, which is an example of *auxiliary learning* paradigm. While multi-task learning aims to improve the performance across all tasks, auxiliary learning differs in that high test accuracy is only required for a primary task, and the role of the other tasks is to assist in generalization of the primary task. Auxiliary learning has been widely used in many areas. In speech recognition, Toshniwal et al. [29] apply auxiliary supervision from phoneme recognition to improve the performance of conversational speech recognition. In computer vision, Liebel at al. [16] propose auxiliary tasks such as the global description of a scene to boost the performance for single scene depth estimation. Mordan et al. [24] observe that object detection can be enhanced if it jointly learns with depth prediction and surface normal prediction as auxiliary tasks. Liu et al. [18] propose a Meta AuXiliary Learning (MAXL) framework that automatically learns appropriate labels for auxiliary tasks. In NLP, Trinh et al. [30] show that unsupervised auxiliary losses significantly improve optimization and generalization of LSTMs. Auxiliary learning has also been applied to improve reinforcement learning [12, 17].

Gradient Direction-based Methods for Adapting Auxiliary Tasks. In auxiliary learning, several methods [7, 17, 34] have been proposed to adapt auxiliary tasks to avoid the situation where they dominate or compete with the target task, where an auxiliary gradient will be down-weighted or masked out if its direction is conflicting with the direction of the target gradient. We will introduce these methods in detail in the Appendix (Section A.1.1) and compare them with the proposed MetaBalance. In particular, MetaBalance does not punish auxiliary gradients with conflict directions but strengths the dominance of the target task from the perspective of gradient magnitudes. In the experiments, MetaBalance shows better generalization than these gradient direction-based methods.

Multi-Task Learning. Multi-task learning [26, 32] is used to improve the learning efficiency and prediction accuracy of multiple tasks via training them jointly. Shared-bottom model [32] is a commonly used structure where task-specific tower networks receive the same representations that come from a shared bottom network.

Multi-Task Balancing Methods. In multi-task learning, methods have been proposed to balance the joint learning of all tasks to avoid the situation where one or more tasks have a dominant influence on the network weight [4, 14, 19, 23, 27]. Although these methods have no special preference to the target task (as in our focus in this paper), we do discuss their connection to MetaBalance in Section A.1.2 (in Appendix) and experimentally compare with them in Section 5.

3 PROBLEM STATEMENT

Our goal is to improve the test accuracy of a target task via training auxiliary tasks alongside this target task on a multi-task network, where useful knowledge from auxiliary tasks can be transferred so that the shared parameters of the network converge to more robust features for the target task. In the context of personalized recommendation, the target task is normally to predict if a user will interact (e.g., purchase or click) with an item, which can be formulated as a binary classification problem. The test accuracy is measured over the top-K items ranked by their probabilities of being interacted with by the user against the ground-truth set of items that the user actually interacted with.

Let θ denote a subset of the shared parameters. For example, θ could be the weight matrix or the bias vector of a multi-layer perceptron in the shared bottom network. θ is learned by jointly minimizing the target task loss $\mathcal{L}_{tar}$ with auxiliary task losses $\mathcal{L}_{aux,i}, i = 1, ..., K$:

$$\mathcal{L}_{total} = \mathcal{L}_{tar} + \sum_{i=1}^{K} \mathcal{L}_{aux,i} \tag{1}$$

We assume that we update θ^t via gradient descent with learning rate α:

$$\theta^{t+1} = \theta^t - \alpha * \mathbf{G}^t_{total} \tag{2}$$

where t means the t-th training iteration over the mini-batches ($t = 1, 2...T$) and $\mathbf{G}^t_{total}$ is the gradient of $\mathcal{L}^t_{total}$ w.r.t θ:

$$\mathbf{G}^t_{total} = \nabla_\theta \mathcal{L}^t_{total} = \nabla_\theta \mathcal{L}^t_{tar} + \sum_{i=1}^{K} \nabla_\theta \mathcal{L}^t_{aux,i} \tag{3}$$

where $\mathbf{G}_{total}$ is equivalent to adding up each gradient of the target and auxiliary losses. To simplify the notations, we have:

- $\mathbf{G}_{tar}$ (i.e., $\nabla_\theta \mathcal{L}_{tar}$): the gradient of the target task loss $\mathcal{L}_{tar}$ with respect to θ.
- $\mathbf{G}_{aux,i}$ (i.e., $\nabla_\theta \mathcal{L}_{aux,i}$): the gradient of the i-th auxiliary task loss $\mathcal{L}_{aux,i}$ with respect to θ, where $i = 1, 2...K$.
- $\|\mathbf{G}\|$: the magnitude (L2 Norm) of the corresponding gradient.

As shown in Eq 3 and 2, the larger the magnitude of a gradient is, the greater the influence this gradient has in updating θ.

4 PROPOSED METHOD

The imbalance of gradient magnitudes may negatively affect the target task optimization. On the one hand, if $\|\mathbf{G}_{aux,i}\|$ ($\exists i \in \{1, 2...K\}$) is much larger than $\|\mathbf{G}_{tar}\|$, the target task will lose its dominance of updating θ and get lower performance. On the other hand, if $\|\mathbf{G}_{aux,i}\|$ ($\exists i \in \{1, 2...K\}$) is much smaller than $\|\mathbf{G}_{tar}\|$, the corresponding auxiliary task might become less influential to assist the target task. As illustrated in Figure 1, many personalized recommendations may suffer from this imbalance. Hence, we are motivated to propose a new algorithm that adapts auxiliary tasks from the perspective of gradient magnitudes.

Algorithm 1 The Basic Version of MetaBalance

Input: θ^1, $\mathcal{L}_{tar}$, $\mathcal{L}_{aux,1}$, ..., $\mathcal{L}_{aux,K}$, ***Strategy*** that is selected from $\{\|\mathbf{G}_{aux,i}\| > \|\mathbf{G}_{tar}\|$, $\|\mathbf{G}_{aux,i}\| < \|\mathbf{G}_{tar}\|$, $(\|\mathbf{G}_{aux,i}\| > \|\mathbf{G}_{tar}\|)$ or $(\|\mathbf{G}_{aux,i}\| < \|\mathbf{G}_{tar}\|)\}$
Output: θ^T

1: **for** t = 1 to T **do**
2: $\quad \mathbf{G}^t_{tar} = \nabla_\theta \mathcal{L}^t_{tar}$
3: $\quad$ **for** i = 1 to K **do**
4: $\quad\quad \mathbf{G}^t_{aux,i} = \nabla_\theta \mathcal{L}^t_{aux,i}$
5: $\quad\quad$ **if** (*Strategy*) **then**
6: $\quad\quad\quad \mathbf{G}^t_{aux,i} \leftarrow \mathbf{G}^t_{aux,i} * \frac{\|\mathbf{G}^t_{tar}\|}{\|\mathbf{G}^t_{aux,i}\|}$
7: $\quad\quad$ **end if**
8: $\quad$ **end for**
9: $\quad \mathbf{G}^t_{total} = \mathbf{G}^t_{tar} + \mathbf{G}^t_{aux,1} + \ldots \mathbf{G}^t_{aux,K}$ (element-wise addition)
10: $\quad$ Update θ using $\mathbf{G}^t_{total}$ (e.g., Gradient Descent: $\theta^{t+1} = \theta^t - \alpha * \mathbf{G}^t_{total}$)
11: **end for**

4.1 Adapting Auxiliary Gradient Magnitudes

As discussed above, the magnitude imbalance between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}, ..., \mathbf{G}_{aux,K}$ may negatively affect the target task optimization. To alleviate this imbalance, MetaBalance is proposed to dynamically and adaptively balance the magnitudes of auxiliary gradients with three strategies and a relax factor (will be detailed in the next subsection).

The basic version of MetaBalance is presented in Algorithm 1, including four steps:

(1) **Calculating the Gradients.** In each training iteration, we firstly calculate $\mathbf{G}^t_{tar}$ and $\mathbf{G}^t_{aux,i}$ respectively (line 2 and 4).
(2) **Applying the Strategy.** In line 5, we can choose either reducing auxiliary gradients with larger magnitudes than the target gradient, or enlarging auxiliary gradients with smaller magnitudes, or applying the two strategies together. The strategy can be selected based on the validation performance of the target task.
(3) **Balancing the Gradients.** Next, $\mathbf{G}^t_{aux,i}$ is normalized to be a unit vector by dividing by $\|\mathbf{G}^t_{aux,i}\|$ and then rescaled to have the same magnitude as $\mathbf{G}^t_{tar}$ by multiplying $\|\mathbf{G}^t_{tar}\|$ (line 6).
(4) **Updating the Parameters.** After that, $\mathbf{G}^t_{total}$ (line 9) is obtained by summing $\mathbf{G}^t_{tar}$ and balanced $\mathbf{G}^t_{aux,1}, \ldots \mathbf{G}^t_{aux,K}$ together. Then, $\mathbf{G}^t_{total}$ is used to update θ following an optimizer's rule such as gradient descent (line 10). Since step (3) and (4) are completely decoupled, MetaBalance has the potential to collaborate with most commonly used optimizers like Adam and Adagrad [8].

MetaBalance benefits auxiliary learning from six aspects:

(1) $\mathbf{G}^t_{aux,i}$ with much larger magnitude than $\mathbf{G}^t_{tar}$ could be automatically reduced, which prevents the dominance of one or more auxiliary tasks for the target task. (Strategy A)
(2) $\mathbf{G}^t_{aux,i}$ with much smaller magnitude than $\mathbf{G}^t_{tar}$ could be automatically enlarged, which enhances the knowledge transference from the corresponding auxiliary task. (Strategy B)
(3) The (1) and (2) could be done together if necessary. (Strategy C)
(4) The strategy is selected based on the target task's performance over validation dataset, which is the empirically best strategy for a specific task and dataset.
(5) Because $\frac{\|\mathbf{G}^t_{tar}\|}{\|\mathbf{G}^t_{aux,i}\|}$ can be regarded as a dynamic weight for $\mathbf{G}^t_{aux,i}$ in line 6, MetaBalance can balance $\mathbf{G}^t_{aux,i}$ dynamically throughout the training process.
(6) As shown in Figure 1, the imbalance of gradient magnitudes varies across the different parts of the same network (e.g., the auxiliary gradients might be much larger than the target gradient in an MLP but much smaller in an embedding layer). Because MetaBalance can be easily applied to each part of the shared parameters separately (θ is an input of Algorithm 1), the training of the different parts can be balanced respectively and adaptively. (5) and (6) makes MetaBalance more flexible than using fixed weights for task losses.

However, the drawback of this basic version in Algorithm 1 is also obvious: forcing auxiliary gradients to have exactly the same magnitude as the target gradient might not be optimal for the target task. To overcome this inflexibility of the magnitude scaling, we design a relax factor to control the closeness of $\|\mathbf{G}^t_{aux,i}\|$ to $\|\mathbf{G}^t_{tar}\|$ in the following subsection.

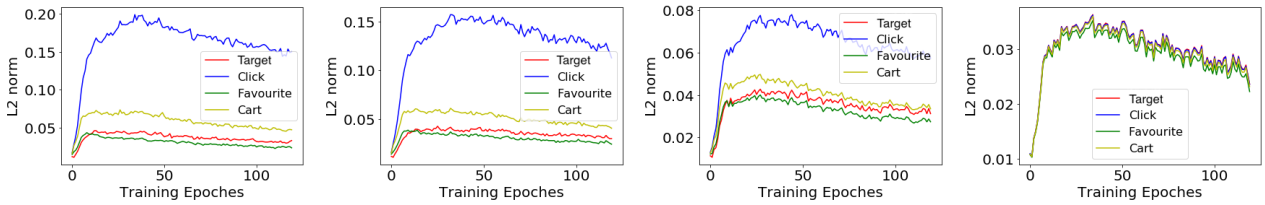
4.2 Adjusting Magnitude Proximity

The next question is how to flexibly adjust the magnitude proximity between $\mathbf{G}_{aux,i}$ and $\mathbf{G}_{tar}$ to adapt to different scenarios? We design a relax factor r to control this magnitude proximity, which is used in line 6 of Algorithm 1:

$$\mathbf{G}^t_{aux,i} \leftarrow (\mathbf{G}^t_{aux,i} * \frac{\|\mathbf{G}^t_{tar}\|}{\|\mathbf{G}^t_{aux,i}\|}) * r + \mathbf{G}^t_{aux,i} * (1 - r)$$

where, if $r = 1$, then $\mathbf{G}^t_{aux,i}$ has exactly the same magnitude as $\mathbf{G}^t_{tar}$. If $r = 0$, then $\mathbf{G}^t_{aux,i}$ keeps its original magnitude. The larger r is, the closer $\|\mathbf{G}^t_{aux,i}\|$ gets to $\|\mathbf{G}^t_{tar}\|$. Hence, r balances the magnitude information between each auxiliary gradient and the target gradient.

The impact of r on the magnitude proximity is illustrated in Figure 2. We observe that the target gradient is dominated by an auxiliary gradient with its much larger magnitude when $r = 0$ in Figure 1(b). In contrast, $r = 1$ lets all gradients have the same but very small magnitude as the target gradient in Figure 2(d). Between the two extremes, Figure 2(b) ($r = 0.2$) and Figure 2(c) ($r = 0.7$) balance the gradient magnitudes in a more moderate way, which



(a) MetaBalance with $r = 0$ (b) MetaBalance with $r = 0.2$ (c) MetaBalance with $r = 0.7$ (d) MetaBalance with $r = 1.0$

Figure 2: The impact of relax factor r on magnitude proximity on UserBehavior-2017 dataset. In the legend, "target" represents the target task (i.e., purchase prediction). Y-axis is the average gradient magnitude over all mini-batch iterations in one epoch, where the gradient w.r.t a MLP layer of the multi-task network is taken as the example.

pushes $\|\mathrm{G}^t_{aux,i}\|$ closer to $\|\mathrm{G}^t_{tar}\|$ but not exactly the same – the original magnitude can be partially kept.

More than that, *r can actually affect the weight for each auxiliary task*. We can further reformulate line 6 in Algorithm 1 as:

$$\mathrm{G}^t_{aux,i} \leftarrow \mathrm{G}^t_{aux,i} * w^t_{aux,i}$$

where $w^t_{aux,i}$ is the weight for $\mathrm{G}^t_{aux,i}$ and we have:

$$w^t_{aux,i} = \left(\frac{\|\mathrm{G}^t_{tar}\|}{\|\mathrm{G}^t_{aux,i}\|} - 1\right) * r + 1 \tag{4}$$

where, if $\|\mathrm{G}^t_{tar}\| > \|\mathrm{G}^t_{aux,i}\|$, the higher r is, the higher $w^t_{aux,i}$ will be; however, if $\|\mathrm{G}^t_{tar}\| < \|\mathrm{G}^t_{aux,i}\|$, the higher r is, the lower $w^t_{aux,i}$ will be.

The next key question is how to choose this r? As presented in Equation 4, r affects the weight for each auxiliary task. Without the prior knowledge of the importance of each auxiliary task to the target task, we treat the setting of r as a data-driven problem and believe that r should be carefully adjusted to adapt to different scenarios. Since r is only used in the backward propagation and hence has no gradient from any loss, r is not a learnable parameter inherently. Hence, we treat r as a hyper-parameter, which is tuned over validation datasets. Note that the same r for all auxiliary tasks does not mean that they will have the same weight or gradient magnitude because $w^t_{aux,i}$ is not only decided by r but also affected by $\|\mathrm{G}^t_{tar}\|$ and $\|\mathrm{G}^t_{aux,i}\|$ (see Equation 4).

Therefore, there is only one hyper-parameter r in MetaBalance that needs to be tuned, which is irrespective of the number of tasks. In contrast, the computational complexity of tuning weights of task losses increases exponentially for each task added. Moreover, we also observe that MetaBalance achieves higher test accuracy than tuning the task weights in our experiments.

Finally, instead of using current magnitudes $\|\mathrm{G}^t_{tar}\|$ and $\|\mathrm{G}^t_{aux,i}\|$ in Algorithm 1, following [23], we apply the moving average of magnitude of the corresponding gradient to take into account the variance among all gradient magnitudes over the training iterations:

$$m^t_{tar} = \beta * m^{t-1}_{tar} + (1-\beta) * \|\mathrm{G}^t_{tar}\| \tag{5}$$

$$m^t_{aux,i} = \beta * m^{t-1}_{aux,i} + (1-\beta) * \|\mathrm{G}^t_{aux,i}\|, \forall i = 1, ..., K \tag{6}$$

where $m^0_{tar} = m^0_{aux,i} = 0$ and β is to control the exponential decay rates of the moving averages, which could be empirically set as 0.9. The moving averages make the training more stable and will be discussed in the experiments. Finally, the complete version of MetaBalance is shown in Algorithm 2.

4.3 Time and Space Complexity Analysis

In this section, we show that MetaBalance does **not** significantly increase the time and space complexity of training multi-task networks. Assume that addition, subtraction, multiplication, division and square root take "one unit" of time. The time complexity of training a multi-task network depends on the network's structure. For simplicity, assume that an MLP is the shared layer of a multi-task network and θ is a weight matrix of a single layer in the MLP, where θ has input dimension n and output dimension m. The time complexity of updating θ is $O(T(1+K)nmd)$ [2], where T is the number of training iterations over the mini-batches, $(1+K)$ is the count of the target task plus K auxiliary tasks and d is the size of the mini-batch. For MetaBalance in Algorithm 2, in each training iteration and for each task, r is a hyper-parameter, calculating $m^t_{aux,i}$ or m^t_{tar} takes $O(nmd)$, and the time complexity of updating the magnitude of $\mathrm{G}^t_{aux,i}$ (line 9) is also $O(nmd)$. To sum up, the time complexity of MetaBalance is still $O(T(1+K)nmd)$. Therefore, MetaBalance will not significantly slow down the training of multi-task networks.

Algorithm 2 The Complete Version of MetaBalance

Input: θ^1, $\mathcal{L}_{tar}$, $\mathcal{L}_{aux,1}$, ..., $\mathcal{L}_{aux,K}$, relax factor r, β in moving averages, ***Strategy*** that is selected from $\{m_{aux,i} > m_{tar}$, $m_{aux,i} < m_{tar}$, $(m_{aux,i} > m_{tar})$ or $(m_{aux,i} < m_{tar})\}$
Output: θ^T

1: Initialize $m^0_{tar} = m^0_{aux,i} = 0$
2: **for** t = 1 to T **do**
3: $\quad \mathrm{G}^t_{tar} = \nabla_\theta \mathcal{L}^t_{tar}$
4: $\quad m^t_{tar} = \beta * m^{t-1}_{tar} + (1-\beta) * \|\mathrm{G}^t_{tar}\|$
5: $\quad$ **for** i = 1 to K **do**
6: $\quad\quad \mathrm{G}^t_{aux,i} = \nabla_\theta \mathcal{L}^t_{aux,i}$
7: $\quad\quad m^t_{aux,i} = \beta * m^{t-1}_{aux,i} + (1-\beta) * \|\mathrm{G}^t_{aux,i}\|$
8: $\quad\quad$ **if** (*Strategy*) **then**
9: $\quad\quad\quad \mathrm{G}^t_{aux,i} \leftarrow (\mathrm{G}^t_{aux,i} * \frac{m^t_{tar}}{m^t_{aux,i}}) * r + \mathrm{G}^t_{aux,i} * (1-r)$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad \mathrm{G}^t_{total} = \mathrm{G}^t_{tar} + \mathrm{G}^t_{aux,1} + \ldots \mathrm{G}^t_{aux,K}$ (element-wise addition)
13: $\quad$ Update θ using G^t_{total} (e.g., $\theta^{t+1} = \theta^t - \alpha * \mathrm{G}^t_{total}$)
14: **end for**

Except for the space of training a multi-task network, MetaBalance only requires extra space for m_{tar}, r, β and $m_{aux,i}$,..., $m_{aux,K}$,

where the space complexity is $O(3+K) = O(1)$ (K is normally a small number). Hence, MetaBalance does not significantly increase the space complexity of multi-task networks training either.

5 EXPERIMENTS

In this section, we present our results and discussion toward answering the following experimental research questions:

- **RQ1:** How well does MetaBalance improve the target task via adapting the magnitudes of auxiliary gradients?
- **RQ2:** How well does MetaBalance perform compared to previous auxiliary task adapting and multi-task balancing methods?
- **RQ3:** How well does MetaBalance collaborate with commonly used optimizers such as Adam and Adagrad?
- **RQ4:** What is the impact of moving averages of gradient magnitudes in MetaBalance?

5.1 Experimental Setup

Following the auxiliary learning setting [18, 31], high test accuracy is only required for a target task while the role of auxiliary tasks is to assist the target task to achieve better test accuracy.

Datasets. IJCAI-2015[2] is a public dataset from IJCAI-15 contest, which contains millions of anonymized users' shopping logs in the past 6 months. UserBehavior-2017[3] is a public dataset of anonymized user behaviors from Alibaba. The two datasets both contain users' behaviors including click, add-to-cart, purchase and add-to-favorite. The statistics of preprocessed datasets are summarized in Table 5 (in Appendix). We treat purchase prediction as the target task and the prediction of other behaviors as auxiliary tasks. We formulate the prediction of each behavior like purchase as a binary classification problem and negative samples are randomly selected.

Evaluation and metrics. In the evaluation, **all items** are ranked according to the probability of being purchased by the user and the top-K items are returned and measured against the ground-truth items set of what users actually purchased, where we adopt three metrics: Normalized Discounted Cumulative Gain (NDCG) [13] at 10 and 20 (N@10 and N@20), precision at 10 and 20 (P@10 and P@20), and recall at 10 and 20 (R@10 and R@20).

Multi-task network. Because how to design a better multi-task network is not the emphasis of this paper, we directly adopt the combination of MLP layer and matrix factorization layer as the shared bottom network, which is widely adopted for recommendations in both academia [10] and industry like Google [6] and Facebook [25]. We build MLP layer as the task-specific tower for each task. The multi-task network is shown in Figure 6 (Appendix).

Baselines. We compare MetaBalance with 10 baseline methods. Gradient direction-based methods that are designed for adapting auxiliary tasks to improve the target task, which will be detailed in Section A.1.1, including: **GradSimilarity** [7], **GradSurgery** [34], **OL-AUX** [17]. Multi-Task balancing methods that treat all tasks equally, which will be detailed in Section A.1.2, including: **Uncertainty** [14], **GradNorm** [4], **DWA** [19], **MTAdam** [23] and **MGDA** [27]. And three simple baselines. **Single-Loss**: we mask out the loss terms of auxiliary tasks and only use target task loss to calculate gradients and update parameters in the model. **Vanilla-Multi**: multiple loss terms are not balanced where the weights for all loss terms are 1. **Weights-Tuning**: weights of loss terms are obtained by random search.

Reproducibility. Due to limited space, the details of reproducibility is presented in Appendix (Section A.2), including dataset preprocessing and split, implementation and training details.

5.2 RQ1: Improvement of Target Task via Adapting Auxiliary Gradients

In this subsection, we discuss the impact of adapting auxiliary gradient magnitudes on the target task's performance.

Impact of Strategy Selection. We firstly study which strategy is optimal for the two recommendation datasets. Note that we firstly compare the three strategies over the validation dataset to choose the best one and apply it on the test dataset. To be consistent with other experimental results, we present the results of the three strategies over the test dataset in Table 1, which reflects the same pattern as the validation dataset. First of all, all three strategies significantly outperform vanilla multi-task learning baseline ("Vanilla-Multi") in UserBehavior-2017 and Strategy C significantly outperforms the baseline in IJCAI-2015, which shows the effectiveness and robustness of MetaBalance. We observe the pattern "Strategy C > Strategy A > Strategy B" across the two datasets, which shows that strengthening the dominance of the target task (Strategy A) is more important than enhancing the knowledge transferring from weak auxiliary tasks (Strategy B) and combining the two strategies together can achieve further improvements for the two datasets. Therefore, we apply Strategy C in the rest of the experiments.

Impact of Relax Factor. Based on Strategy C, we further study the impact of the relax factor. Figure 3 presents the varying of NDCG@10 and Recall@10 as r changes in UserBehavior-2017 dataset (the similar observation is obtained in IJCAI-2015 dataset).

The worst NDCG@10 and Recall@10 are achieved when $r = 0$ (Vanilla-Multi), where auxiliary gradients ($\|\mathbf{G}^t_{aux,i}\|$) keep their original magnitudes (i.e., not balanced as in Vanilla-Multi). In Figure 1(b), we observe that click task's gradient magnitude (blue curve) is much larger than $\|\mathbf{G}^t_{tar}\|$ (red curve). Hence, the target task gradient is probably dominated by the click task gradient, which explains the low target task performance of Vanilla-Multi (see Table 2).

In contrast, $r = 1$ in MetaBalance means that $\|\mathbf{G}^t_{aux,i}\|$ becomes very close to $\|\mathbf{G}^t_{tar}\|$ as shown in Figure 2(d), where the four curves are twisted together. However, $r = 1$ achieves suboptimal results as shown in Figure 3, which demonstrates that the target task performance might be negatively impacted by a large r. A possible reason is that most auxiliary gradients are reduced to be very similar to the target gradient and hence the update of the shared parameters becomes so small that it negatively affects the optimization.

Between the two extremes, Figure 2(b) ($r = 0.2$) and Figure 2(c) ($r = 0.7$) balance the gradient magnitudes in a more moderate way – getting $\|\mathbf{G}^t_{aux,i}\|$ closer to $\|\mathbf{G}^t_{tar}\|$ but not exactly the same, where $r = 0.7$ achieves the best performance as shown in Figure 3.

[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=47&userId=1
[3] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

Table 1: Strategy Selection

Datasets	UserBehavior-2017			IJCAI-2015		
Metrics (%)	N@10	R@10	P@10	N@10	R@10	P@10
Vanilla-Multi	0.820	1.284	0.291	0.844	0.965	0.437
Strategy A (strengthening the dominance of the target task)	0.948	1.487	0.316	0.858	0.963	0.424
Strategy B (enhancing the knowledge transferring from weak auxiliary tasks)	0.904	1.384	0.301	0.818	0.950	0.425
Strategy C (Adopting Strategy A and Strategy B together)	0.990	1.550	0.339	0.974	1.164	0.509

Table 2: Experimental Results

	UserBehavior-2017						IJCAI-2015					
Metric(%)	N@10	R@10	P@10	N@20	R@20	P@20	N@10	R@10	P@10	N@20	R@20	P@20
Single-Loss	0.817	1.265	0.275	0.994	1.825	0.208	0.883	0.935	0.431	1.022	1.314	0.298
Vanilla-Multi	0.820	1.284	0.291	1.074	2.107	0.237	0.844	0.965	0.437	0.992	1.353	0.311
Weights-Tuning	0.909	1.378	0.326	1.165	2.195	0.263	0.866	1.013	0.445	1.037	1.448	0.330
Uncertainty	0.724	1.158	0.266	0.903	1.739	0.201	0.695	0.818	0.365	0.834	1.186	0.266
GradNorm	0.913	1.292	0.297	1.147	2.044	0.237	0.878	0.953	0.430	1.035	1.375	0.307
DWA	0.915	1.419	0.309	1.165	2.232	0.248	0.899	1.005	0.442	1.040	1.372	0.312
MGDA	0.845	1.328	0.292	1.075	2.058	0.237	0.809	1.104	0.439	1.104	1.673	0.350
MTAdam	0.869	1.382	0.305	1.112	2.153	0.247	0.880	1.015	0.463	1.071	1.525	0.348
GradSimilarity	0.923	1.444	0.308	1.186	2.270	0.255	0.817	0.977	0.427	1.025	1.529	0.336
GradSurgery	0.936	1.471	0.319	1.213	2.371	0.263	0.876	0.998	0.445	1.042	1.434	0.327
OL-AUX	0.931	1.471	0.311	1.162	2.224	0.243	0.804	0.921	0.413	0.950	1.312	0.295
MetaBalance	**0.990***	**1.550***	**0.339***	**1.258***	**2.421***	**0.269***	**0.974***	**1.164***	**0.509***	**1.134***	**1.588**	**0.353**
Improvement	5.77%	5.32%	3.96%	3.66%	2.09%	2.08%	8.34%	14.68%	10.01%	2.72%	–	0.86%

∗ We conduct a two-sided significant test between MetaBalance and the strongest baseline (highlighted by underscore), where * means the p-value is smaller than 0.05.

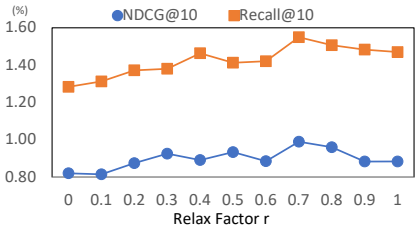


Figure 3: Impact of relax factor r

5.3 RQ2: Comparison with Baseline Methods

Table 2 presents the experimental results and the improvement of MetaBalance upon the strongest baseline in terms of each metric, where MetaBalance significantly outperforms all baselines over most of metrics on the two datasets.

MetaBalance vs. gradient direction-based methods. First, we observe that MetaBalance outperforms GradSimilarity, OL-AUX and GradSurgery, which are designed to boost the target task via adapting auxiliary tasks. Remember that the same idea behind these methods is that the less similar the direction of target gradient and one auxiliary gradient is, the lower weight will be assigned to that auxiliary task. While these gradient direction-based methods have worse performance than MetaBalance over the testing dataset, interestingly, they actually achieve better **training** loss than MetaBalance, where an example is shown in Figure 4, which demonstrates they are more prone to overfitting than MetaBalance. Hence, this observation reveals that auxiliary gradients that have dissimilar directions with the target gradient might be sometimes helpful to improve the generalization ability of the model, which is consistent with the observations in the literature [5, 32]. For example, they might help the target task to correct its direction of the optimization to achieve a better generalization ability. MetaBalance keeps the direction conflicts between the target gradient and the auxiliary gradients but reduces the auxiliary gradients whose magnitudes are much larger than the target gradient, which prevents the dominance of auxiliary tasks and shows more robust performance for personalized recommendations.

In addition, we are curious if MetaBalance can be enhanced when it also considers using the direction similarity to adapt auxiliary gradients. Specifically, in each training iteration, we first enlarge or reduce auxiliary gradients via MetaBalance and then enlarge or reduce them again according to one of the gradient direction-based methods. The results in Table 3 show that the performance of MetaBalance mostly drops after including the gradient direction-based methods, which demonstrates that naively combining both magnitude and direction-based approaches can interfere with each another. We leave how to better consider both gradient magnitudes and directions for adapting auxiliary tasks to help the target task in the future work.

MetaBalance vs. multi-task balancing methods. Second, it is understandable that Uncertainty, GradNorm, DWA are inferior to MetaBalance because they have no special preference to the target task. In DWA, the lower the loss decreases, the higher the weight is assigned to that loss. In GradNorm, the target task gradient magnitude is regularized to be similar to the average of all gradient

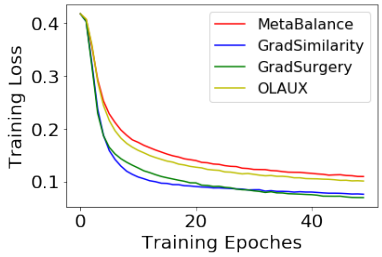


Figure 4: The training loss on UserBehavior-2017

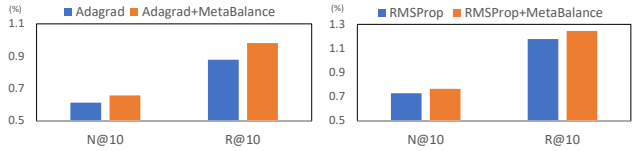
magnitudes, which might not be the optimal magnitude for the target task optimization. In Uncertainty, the higher the uncertainty of the task dataset, the higher weight is assigned to that task loss. We also compare MGDA [27] as one of the most representative Pareto methods with MetaBalance. MGDA treats multi-task learning as multi-objective optimization problem and finds solutions that satisfy Pareto optimality. In MGDA, the shared parameters are only updated along common directions of the gradients for all tasks, which might not be the best optimization direction for the target task. Consequently, the target task is not guaranteed to be improved the most among all tasks in Pareto optimal solutions like MGDA. In contrast, MetaBalance is a specialized method designed for boosting the target task. As Table 2 shows, MetaBalance significantly outperforms MGDA over most of the metrics. Although MTAdam is not originally designed for auxiliary learning, we let the target task serve as the anchor task in MTAdam. In this way, MetaBalance and MTAdam share the same core idea that the auxiliary gradient magnitudes become closer to the target gradient. However, Table 2 shows that MetaBalance significantly outperforms MTAdam. The possible reason might be the relax factor in MetaBalance that can control the magnitude proximity, which makes MetaBalance more flexible than MTAdam.

In addition, Vanilla-Multi is even inferior to Single-loss over most of metrics on both datasets. This demonstrates that transfer learning from auxiliary tasks is a non-trivial task – that might hurt the performance of the target task rather than boosting it. After that, Table 2 shows that Weights-Tuning, where the target task loss normally has a higher weight assigned than the auxiliary tasks, outperforms Vanilla-Multi over all metrics on both datasets. However, the performance of Weights-Tuning is significantly inferior to MetaBalance. A possible reason is that the tuned weights are fixed during the training and hence behave sub-optimally in adapting auxiliary tasks.

To sum up, the results demonstrate that the three strategies and the relax factor make MetaBalance a flexible and effective framework to adapt auxiliary tasks from the perspective of gradient magnitudes, which significantly improves the target task's performance and outperforms baselines.

Table 3: MetaBalance plus Gradient Direction-based Methods

Metric(%)	N@10	R@10	P@10	N@20	R@20	P@20
MetaBalance (MB)	0.990	1.550	0.339	1.258	2.421	0.269
MB+GradientSimilarity	0.937	1.398	0.311	1.190	2.210	0.250
MB+GradientSurgery	0.925	1.585	0.329	1.167	2.351	0.258
MB+OL-AUX	0.898	1.374	0.308	1.158	2.224	0.248



(a) Adagrad (b) RMSProp

Figure 5: Collaboration with Other Optimizers

5.4 RQ3: Collaboration with More Optimizers

As shown in Algorithm 1 and 2, MetaBalance balances the gradient magnitudes and these balanced gradients are used to update shared parameters following the rules of optimizers. Results in Table 2 have shown that MetaBalance can collaborate with Adam well. We are also curious if MetaBalance can collaborate with other popular optimizers – achieving higher performance for the target task compared to the multi-task network that is trained without MetaBalance. In Figure 5, we observe that two other widely used optimizers – Adagrad [8] and RMSProp [28] – can also achieve better performance via using the balanced gradients from MetaBalance. This result demonstrates that MetaBalance can flexibly collaborate with commonly-used optimizers.

5.5 RQ4: Impact of Moving Averages of Gradient Magnitudes

In Table 4, we compare the performance of MetaBalance with its variant ("–Moving Average") where the moving averages of magnitude m^t_{tar} and $m^t_{aux,i}$ (in Equation 5 and 6) are replaced with the current magnitudes $\mathbf{G}^t_{tar}$ and $\mathbf{G}^t_{aux,i}$ at each iteration. We observe that the performance drops slightly on UserBehavior-2017 and drastically on IJCAI-2015 dataset. This result demonstrates the moving averages of magnitudes benefits the optimization, which takes into account the variance among all gradient magnitudes over the training iterations.

Table 4: Ablation Study of Moving Average of Magnitude

Datasets	UserBehavior-2017			IJCAI-2015		
Metrics (%)	N@10	R@10	P@10	N@10	R@10	P@10
MetaBalance	0.990	1.550	0.339	0.974	1.164	0.509
–Moving Average	0.983	1.513	0.325	0.835	0.956	0.426

6 CONCLUSION

In many personalized recommendation scenarios, the target task can be improved via training auxiliary tasks alongside this target task on a multi-task network. In this paper, we propose MetaBalance to adapt auxiliary tasks to better assist the target task from the perspective of gradient magnitude. Specifically, MetaBalance has three adapting strategies, such that it not only protects the target task from the dominance of auxiliary tasks but also avoids that one or more auxiliary tasks are ignored. Moreover, auxiliary gradients are balanced dynamically throughout the training and adaptively for each part of the network. Our experiments show that MetaBalance can be flexibly adapted to different scenarios and significantly outperforms previous methods on two real-world datasets.

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *proceedings of the 10th ACM Conference on Recommender Systems*. 107–114.

[2] Walter Baur and Volker Strassen. 1983. The complexity of partial derivatives. *Theoretical computer science* 22, 3 (1983), 317–330.

[3] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*. PMLR, 794–803.

[5] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *arXiv preprint arXiv:2010.06808* (2020).

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

[7] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. 2018. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224* (2018).

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

[9] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[11] Yun He, Jianling Wang, Wei Niu, and James Caverlee. 2019. A Hierarchical Self-Attentive Model for Recommending User-Generated Item Lists. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1481–1490.

[12] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* (2016).

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* (2002).

[14] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

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

[16] Lukas Liebel and Marco Körner. 2018. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334* (2018).

[17] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. 2019. Adaptive Auxiliary Task Weighting for Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 4772–4783.

[18] Shikun Liu, Andrew Davison, and Edward Johns. 2019. Self-supervised generalisation with meta auxiliary learning. In *Advances in Neural Information Processing Systems*. 1679–1689.

[19] 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.

[20] Yidan Liu, Min Xie, and Laks VS Lakshmanan. 2014. Recommending user generated item lists. In *Recsys*.

[21] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 931–940.

[22] 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.

[23] Itzik Malkiel and Lior Wolf. 2020. MTAdam: Automatic Balancing of Multiple Training Loss Terms. *arXiv preprint arXiv:2006.14683* (2020).

[24] Taylor Mordan, Nicolas Thome, Gilles Henaff, and Matthieu Cord. 2018. Revisiting multi-task learning with rock: a deep residual auxiliary block for visual detection. *Advances in neural information processing systems* 31 (2018), 1310–1322.

[25] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

[26] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* (2017), arXiv–1706.

[27] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650* (2018).

[28] T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

[29] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. 2017. Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition. *Proc. Interspeech 2017* (2017), 3532–3536.

[30] Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. 2018. Learning Longer-term Dependencies in RNNs with Auxiliary Losses. In *International Conference on Machine Learning*. 4965–4974.

[31] Abhinav Valada, Noha Radwan, and Wolfram Burgard. 2018. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6939–6946.

[32] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. 2020. Multi-Task Learning for Dense Prediction Tasks: A Survey. *arXiv preprint arXiv:2004.13379* (2020).

[33] Xin Wang, Wenwu Zhu, and Chenghao Liu. 2019. Social recommendation with optimal limited attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1518–1527.

[34] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782* (2020).

[35] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative multi-Level embedding learning from reviews for rating prediction.. In *IJCAI*, Vol. 16. 2986–2992.

A APPENDIX

A.1 Relations of MetaBalance to Previous Methods

In this section, we compare MetaBalance with previous methods.

A.1.1 Auxiliary Task Adapting Methods. These methods are specifically designed for auxiliary learning such that auxiliary tasks are adapted to better improve the target task. The common idea is that if $\mathbf{G}_{aux,i}$ is conflicting with $\mathbf{G}_{tar}$ from the perspective of direction, $\mathbf{G}_{aux,i}$ will be down-weighted or masked out. Compared with them, MetaBalance is the first method that adapts auxiliary task to assist the target task from the perspective of gradient magnitudes rather than punishing $\mathbf{G}_{aux,i}$ due to conflicting directions. The experimental results show that keeping inner-competition between the target gradient and conflicting auxiliary gradients as MetaBalance does improves the generalization ability of the model.

GradSimilarity [7] adapts auxiliary tasks via the gradient similarity between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}$. Specifically, if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative, then $\mathbf{G}_{aux,i}$ will not be added to $\mathbf{G}_{total}$ and hence will be ignored in updating the shared layers.

GradSurgery[4] [34] replaces $\mathbf{G}_{aux,i}$ by its projection onto the normal plane of $\mathbf{G}_{tar}$ if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative, unlike [7] where $\mathbf{G}_{aux,i}$ is just ignored. Formally, if $cosine(\mathbf{G}_{tar}, \mathbf{G}_{aux,i})$ is negative, they let:

$$\mathbf{G}^t_{aux,i} = \mathbf{G}^t_{aux,i} - \frac{\mathbf{G}^t_{aux,i} \cdot \mathbf{G}^t_{tar}}{\|\mathbf{G}^t_{tar}\|^2} \cdot \mathbf{G}^t_{tar} \tag{7}$$

In this way, the conflict between $\mathbf{G}_{tar}$ and $\mathbf{G}_{aux,i}$ can be alleviated.

OL-AUX (Oline learning for Auxiliary Losses) [17] defines the total loss as $\mathcal{L}^t = \mathcal{L}^t_{tar} + \sum_{i=0}^{K} w^t_{aux,i} \cdot \mathcal{L}^t_{aux,i}$ where $w_{aux,i}$ is the weight of $\mathcal{L}_{aux,i}$ and $\mathcal{V}^t(\mathbf{w})$ as the speed at which the target task loss decreases at the t-th iteration and $\mathbf{w} = [w_1, ..., w_K]^T$. OL-AUX seek to optimize the N-step decrease of the target task w.r.t $\mathbf{w}$:

$$\mathcal{V}^{t,t+N}(\mathbf{w}) = \mathcal{L}^{t+N}_{tar} - \mathcal{L}^t_{tar} \tag{8}$$

With some approximations, they find that $\forall i = 1, ..., K$:

$$\nabla_{w_{aux,i}} \mathcal{V}^{t,t+N}(w_{aux,i}) = -\sum_{j=0}^{N-1} (\mathbf{G}^{t+j}_{tar})^T \mathbf{G}^{t+j}_{aux,i} \tag{9}$$

Then, $\mathbf{w} \leftarrow \mathbf{w} - \beta \cdot \nabla_{\mathbf{w}} \mathcal{V}^{t,t+N}(\mathbf{w})$ such that the speed at which $\mathcal{L}_{tar}$ decreases could be maximized.

A.1.2 Multi-Task Balancing Methods. In contrast to the auxiliary learning-specific methods, these multi-task balancing methods are for general learning where all tasks are treated equally important. Although they have no preference to the target task, they are valid baselines because MetaBalance is specifically for auxiliary learning and is supposed to outperform them. For convenience, we let j denotes the index of any task in this subsection, where $j = 0, 1, ..., K$. Let w_j be the weight of the j-th task loss $\mathcal{L}_j$.

Uncertainty [14] assumes that the higher the uncertainty of task data is, the lower the weight of this task loss should be assigned. They design a learnable parameter σ_j to model the uncertainty for each task. Specifically, they optimize the model parameters and σ_j to minimize the following objective:

$$\mathcal{L} = \sum_{j=0}^{K} \frac{1}{\sigma_j^2} \mathcal{L}_j + \sum_{j=0}^{K} log\sigma_j \tag{10}$$

Minimizing the loss $\mathcal{L}$ w.r.t. σ_j can automatically balance $\mathcal{L}_j$ during training, where increasing σ_j reduces the weight for task loss $\mathcal{L}_j$.

GradNorm [4] encourages $\|\mathbf{G}^t_j\|$ to be the mean of all $\|\mathbf{G}^t_j\|$, $j = 0, ..., K$. In this way, all tasks could have a similar impact on the updating of shared-parameters. In particular, they minimize the following two objectives:

$$\mathcal{L}^t = \sum_{j=0}^{K} w^t_j \cdot \mathcal{L}^t_j \tag{11}$$

$$\mathcal{L}^t_{normLoss} = \sum_{j=0}^{K} L1Norm(\|w^t_j \cdot \mathbf{G}^t_j\| - \|\overline{\mathbf{G}^t}\| \cdot [r^t_j]^\alpha) \tag{12}$$

In each iteration, $\mathcal{L}^t$ is firstly optimized w.r.t model parameters θ (not including w^t_j) to obtain $\mathbf{G}^t_j$ and the $\mathcal{L}^t_{normLoss}$ is optimized w.r.t w^t_j. In the next iteration, updated w^t_j can balance $\mathcal{L}^t_j$. Moreover, r^t_j is to model the pace at which different tasks are learned, where $r^t_j = p^t_j / E[p^t_j]$ and $p^t_j = \mathcal{L}^t_j / \mathcal{L}^0_j$. And α is a hyper-parameter which sets the strength of forcing tasks back to a common training rate.

DWA (Dynamic Weight Averaging) [19] balances the pace at which tasks are learned. In DWA, w^t_j is set as:

$$w^t_j = \frac{N \cdot exp(p^{t-1}_j / T)}{\sum_n exp(p^{t-1}_n / T)}, \quad p^{t-1}_j = \frac{\mathcal{L}^{t-1}_j}{\mathcal{L}^{t-2}_j} \tag{13}$$

where N is the number of tasks and temperature T controls the softness of the task weighting in the softmax function. p_j estimates the relative descending rate of $\mathcal{L}_j$. When $\mathcal{L}_j$ decreases at a slower rate compared with other task losses, w_j will be increased.

MGDA (Multiple-Gradient Descent Algorithm) [27] treats multi-task learning as multi-objective optimization problem and finds solutions that satisfies Pareto optimality – as long as there is a common direction along which losses can be decreased, we have not reached a Pareto optimal point yet. Since the shared parameters are only updated along **common** directions of the task-specific gradients, MGDA has no preference on a particular task.

MTAdam [23] is an Adam-based optimizer that balances gradient magnitudes and then update parameters according to the rule of Adam [15]. Following MTAdam, we also directly manipulate the gradient magnitudes, instead of weighting task losses like Uncertainty [14], GradNorm [4] and DWA [19]. MetaBalance differs from MTAdam in the following aspects:

- MTAdam lets all gradient magnitudes be similar to that of the first loss (not necessarily the target loss) while MetaBalance has three strategies that can flexibly encourage auxiliary gradients to better help the target task optimization.
- $\|\mathbf{G}^t_{aux,i}\|$ can only be very similar to $\|\mathbf{G}^t_{tar}\|$ in MTAdam while MetaBalance can adjust the proximity of $\|\mathbf{G}^t_{aux,i}\|$ to $\|\mathbf{G}^t_{tar}\|$

[4]GradSurgery is originally for balancing multi-task learning. We can easily apply GradSurgery for auxiliary learning by specifying a task as the target task and the others as the auxiliary tasks

Table 5: Statistics of Preprocessed Datasets

			Target Task		Auxiliary Tasks		
Dataset	#User	#Item	#Buy	Density of Buy	#Add-to-Cart	#Click	#Add-to-Favorite
IJCAI-2015	19,839	50,973	390,600	0.039%	1,693	2,025,910	224,279
UserBehavior-2017	16,089	25,813	89,404	0.022%	53,245	394,246	19,585

via the relax factor, which is vital for adapting MetaBalance to different scenarios.
- MetaBalance is an auxiliary task adapting algorithm that can collaborate with most optimizers like Adagrad or RMSprop to update parameters, whereas MTAdam is specially designed for Adam-based optimizers only.

A.2 Reproducibility of Experiments

The code of our approach can be found at here.[5]

Dataset Preprocessed and Split. IJCAI-2015 is preprocessed by filtering out users who purchase fewer than 20 unique items and items which are purchased by fewer than 10 unique users. We omit add-to-cart as an auxiliary task in IJCAI-2015 because this behavior only has 1,693 feedbacks. For UserBehavior-2017, we filter out users who purchase fewer than 10 unique items and items which are purchased by fewer than 10 unique users. The datasets are summarized in Table 5. We randomly split purchase interactions into a training set (70%), validation set (10%) and testing set (20%). For the interactions of auxiliary tasks like add-to-cart, we merge them into the training set. Since auxiliary interactions like add-to-cart are highly related to purchase interaction, to prevent possible information leakage, we remove user-item pairs from the auxiliary interactions if these pairs appear in the validation set and testing set of the purchase interactions.

Implementation and Training Details. We implement MetaBalance, Uncertainty, DWA, GradSimilarity, GradSurgery and OL-AUX via Pytorch. The code of GradNorm is from this repo[6] and the code of MTAdam is from the authors.[7] All experiments are conducted on an Nvidia GeForce GTX Titan X GPU with 12 GB memory. Cross-entropy loss is adopted for each task and Adam [15] is the optimizer with batch size of 256 and learning rate of 0.001.

Hyper-parameters. All hyper-parameters are carefully tuned in the validation set, where early stopping strategy is applied such that we terminate training if validation performance does not improve over 20 epochs. In the multi-task recommendation network, the size of user and item embeddings is 64, the size of the shared MLP layers is {32, 16, 8} and the size of the task-specific MLP layers is {64, 32}. To prevent overfitting, dropout with rate of 0.5 is applied for each layer and we also use weight decay with rate of e-7. For MetaBalance, r is selected from 0.1, 0.2, ...0.9 and 0.7 is the best for UserBehavior-2017 and 0.9 is the best for IJCAI-2015. For MTAdam, β_1, β_2, β_3 are respectively set as 0.9, 0.999 and 0.9. For DWA, T is set as 2 and we calculate the mean of losses in very 5 iterations on IJCAI-2015 and in very 10 iterations on UserBehavior-2017. For GradNorm, α is set as 0.75 on IJCAI-2015 and 0 on UserBehavior-2017. For OL-AUX, β is set as 0.1 on IJCAI-2015 and 1 on UserBehavior-2017.

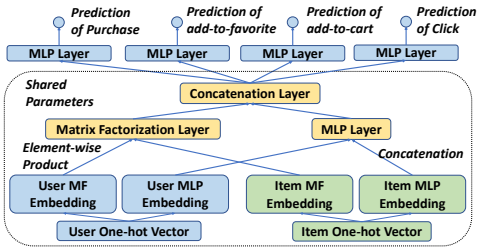


Figure 6: Multi-Task recommendation network in the evaluation. The shared bottom layers is the combination of MLP layer and matrix factorization layer, which is widely adopted for recsys in both academia [10] and industry [6, 25]

[5]https://github.com/facebookresearch/MetaBalance
[6]https://github.com/hosseinshn/GradNorm
[7]https://github.com/ItzikMalkiel/MTAdam