An Online Multi-task Learning Framework for Google Feed Ads Auction Models

Ning Ma
ninm@google.com
Google Inc.
Mountain View, USA

Mustafa Ispir
ispir@google.com
Google Inc.
Mountain View, USA

Yuan Li
yuanliy@google.com
Google Inc.
Mountain View, USA

Yongpeng Yang
yongpeng@google.com
Google Inc.
Mountain View, USA

Zhe Chen
chenzhe@google.com
Google Inc.
Mountain View, USA

Derek Zhiyuan Cheng
zcheng@google.com
Google Inc.
Mountain View, USA

Lan Nie
lannie@google.com
Google Inc.
Mountain View, USA

Kishor Barman
kishorbarman@google.com
Google Inc.
Mountain View, USA

ABSTRACT

In this paper, we introduce a large scale online multi-task deep learning framework for modeling multiple feed ads auction prediction tasks on an industry-scale feed ads recommendation platform. Multiple prediction tasks are combined into one single model which is continuously trained on real time new ads data. Multi-tasking ads auction models in real-time faces many real-world challenges. For example, each task may be trained on different set of training data; the labels of different tasks may have different arrival time due to label delay; different tasks will interact with each other; combining the losses of each task is non-trivial. We tackle these challenges using practical and novel techniques such as multi-stage training for handling label delay, Multi-gate Mixture-of-Experts (MMoE) to optimize model interaction and an auto-parameter learning algorithm to optimize the loss weights of different tasks. We demonstrate that our proposed techniques can lead to quality improvements and substantial resource saving compared to modeling each single task independently.

CCS CONCEPTS

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

KEYWORDS

Computational advertising; Online advertising; Multi-task learning; Recommender systems

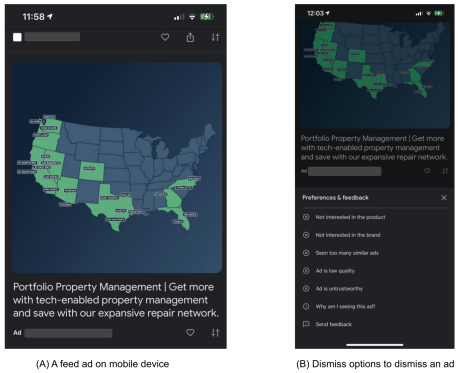
ACM Reference Format:
Ning Ma, Mustafa Ispir, Yuan Li, Yongpeng Yang, Zhe Chen, Derek Zhiyuan Cheng, Lan Nie, and Kishor Barman. 2022. An Online Multi-task Learning Framework for Google Feed Ads Auction Models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539055

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539055

1 INTRODUCTION

Like many other large scale recommendation systems [6], ads recommendation system usually consists of two main stages - candidate generation and ranking. During the candidate generation stage, we retrieve a subset of candidates from a larger corpus. In the ranking stage, the ranking system provides a score for each candidate and generates the final candidates list shown to the users. The ads ranking stage is also referred to as ads auction where the system runs an auction and assigns a score to each ad candidate. The auction score is usually computed by a "bid" and a few predicted quality scores, such as predicted click through rate (pCTR). The "bid" here stands for how much an advertiser is willing to pay when the ad is clicked.



(A) A feed ad on mobile device (B) Dismiss options to dismiss an ad

Figure 1: (A) An ad on Feed. (B) The options to dismiss an ad

Figure 1 (A) shows how a feed ad card is recommended to the users on the Google feed ads platform at some fixed positions. Some important user actions on the ads will be logged such as the click

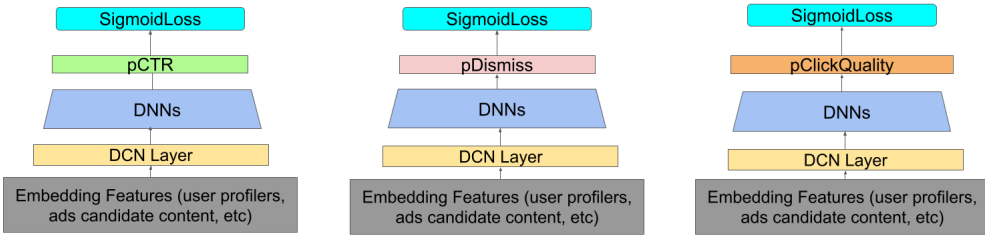


Figure 2: Three independent auction models: pCTR, pDismiss, pClickQuality

action. User can also dismiss a feed ad by clicking one of the dismiss options as shown in the Figure 1 (B). Sophisticated machine learning models are built to predict those user actions. For example, a widely known predicted click through rate (pCTR) model is used to predict the click through rate of an ad. Similarly, we can build models to predict the probability that an ad is not relevant and thus could be dismissed by the user (pDismiss), and the probability that an ad click leads to a good landing page experience (pClickQuality). Then, an auction score can be computed via a formula like Eq. 1 which is a function of those prediction scores and the bid.

$$AuctionScore = f(bid, pCTR, pDismiss, pClickQuality) \quad (1)$$

The AuctionScore can be used to rank and select the ad candidates shown to the users. These different prediction models are trained independently as shown in Figure 2.

In this paper, we propose a large scale online multi-task learning framework for modeling multiple auction prediction tasks in one shot on an industrial feed ads recommendation platform. One single model will perform multiple predictions simultaneously and is continuously trained on real time new data. Unlike a general machine learning model which is repeatedly trained on a fixed training set until convergence and then tested on a fixed validation data set before serving, the online model is trained on each data event once and is continuously trained on incoming data to guarantee the model freshness.

Multi-task learning can provide great benefits including:

- Multi-task will help us measure the unified impact of model changes. For example, whenever we add a new feature it will be automatically incorporated to all predictions.
- We will not have a separate launch for each model. Instead, we have a single launch for a combination of all models. This will increase the engineering velocity drastically.
- We can do 'transfer learning' from frequent labels to sparse ones. Sharing embeddings and hidden layers, we can let sparse labels leverage the information learned by more frequent labels.
- We can save the model resource usage as all tasks will share a certain portion of a single model

If a multi-task model has the same quality performance as individual models, the multi-task model will easily outperform individually trained model due to significant resource saving and engineering velocity improvement.However, despite the huge benefits the multi-task learning can potentially bring, designing and implementing such a large scale online multi-task learning model for real online ads advertising face a lot of unique challengings, including but not limited to

- Each task may need to be trained on different set of training data. For example, pCTR model is trained based on viewed ads. But, pClickQuality is trained based on the clicked ads.
- In a real time online training system, the labels of different tasks may have different arrival time due to different label delay. Click label may arrive in a few hours but some post-click label may not be available in a few days.
- Different tasks will interact with each other which may lead to worse prediction results when compared to the individual models.
- Combining the losses of each task is non-trivial. Different tasks have different label sparsity and different impact on the final auction stage. Bad loss weights may lead to drastic performance drop of one task.

Without handling all those challenges correctly, we will end up with a multi-task model which may perform worse than the original task, leading to revenue drop and/or worse user experience. That's the primary reason why consolidating multiple individual serving tasks into a multi-task framework on the online training system for online advertising is very challenging despite the concept of multi-tasking has been widely studied in the research community. In this paper, we demonstrate the novel and practical techniques we use to tackle those challenges and lead to a few successful launches. In this paper, we focus on consolidating three prediction serving tasks pCTR, pDismiss and pClickQuality, but the techniques can be scaled to arbitrary number of ads prediction task. Our main contributions are the following

- We introduce a large scale online multi-task deep learning framework for modeling multiple feed ads auction prediction tasks with a clean and effective engineering solution
- We designed a multi-stage training method to make sure each task uses the correct labels with different delay time
- We extend the Multi-gate Mixture-of-Experts architecture on shared-bottle structure to learn task interactions which is easy to scale and maintain
- We properly assign different training events to different training task in real time by attributing the loss from a specific training event to a corresponding task
- We propose a novel and practical approach for optimizing the loss weight of different ads quality prediction tasks

In our framework, we use a MMoE representation to enforce sparsity of solution and at the same time encode the delay in each data stream by a different expert to cleanly model the training data delay. We define a novel single multi-task objective which can be easily extended to an auto-parameter tuning service for task loss weight tuning. To evaluate our proposed ranking system, we conduct offline and online live experiments to verify that the proposed multi-task framework works better than the original individual models. We show model quality improvement and resource saving by using our proposed framework. Additionally, the advantage of having each component in the model. e.g., MMOE, training delay and weight optimization, is shown empirically and incrementally.

The rest of the paper is organized as the following: In Section 2, we describe related work on multi-tasking learning and ads recommendation system. In Section 3, we provide the problem description for ads quality prediction, auction score generation and ads ranking. In Section 4, which is the most essential part of this paper, we describe the online learning framework and various techniques we use to tackle the multi-tasking challenges we mentioned previously. In Section 5, we demonstrate the offline and online metrics we use to evaluate our multi-task framework. In section 6, we summarize our finding and discuss future direction.

2 RELATED WORK

Deep neural network based multitask learning has been an active topic in many academic research areas. For example, researchers have been trying to build multi-task learning framework for natural language processing [5] and computer vision tasks [10]. More recently, researchers start working on multi-modality multi-task model for vision-language models [11]. Those academic multi-task frameworks usually iterate a shared model by repeatedly training it on static training data until all tasks converge. In [11], they proposed to use some alogrithms such as 'Round-Robin Batch-Level Sampling' and 'Dynamic Stop-and-Go' to make the multi-taks model converge on each task better. However, all those multi-task models are trained on a static academic training data set and do not have unique challenges a large-scale multi-task learning model will encounter on a real time online ads advertising platform. Unlike traditional offline machine learning model training, online learning models never fully converge on any data set because it is only allowed the model to see each training event once. In the online training framework, the model is able to keep updated with the most recent changes happening in the world, and thus guarantee the model freshness which is critical for real time ads advertising. The online learning has been used in real internet companies and handles the non-stationarity and freshness of the data very well [13].

A lot of research works focus on understanding the relationship between multi-task learning and multi-objective optimization. For example, recent work [16] proposed an algorithm to decide which layers to share for which tasks during the training process. Similarly, [15] proposed another algorithm to decide which tasks should be learned together. Regarding the multi-task multi-objective optimization, some proposed heuristics based on uncertainty and gradient to learn the loss weights of different tasks [3, 8]. Others proposed novel multi-objective optimization algorithms to get pareto optimal solution for multi-task learning [9, 14]. Despite these algorithms are technically sounds, we found them not very practical in building a new online multi-task learning models for online advertising due to the following: 1. Those methodologies do not always fit the objective of online advertising; 2. They may introduce unnecessary engineering complexity while not addressing the fundamental challenges we are facing; 3. Some of them rely on heuristics and thus make the multi-task model not easy to scale and maintain. Nevertheless, we think these novel research ideas could be great follow-up work after a solid online multi-task learning framework has been built for online advertising.

Recently, multi-tasks are also being adopted to some real industrial products. In [22], they built a multi-task ranking system to recommending what Youtube video to watch next. Nevertheless, it is not an online learning for advertising where they mainly focused on extending the structure to a multi-gated mixture of experts (MMoE) [12] and how to address the position and selection biases. Pinterest ads also posted a few research blogs talking about how they built a multi-task framework for Ads auction [17]. However, majority of the techniques they talked about, such as 'negative sampling', 'feature processing' and ' bias correction', are not specific to multi-tasking. The main techniques related to the multi-tasking is the multi-tower structure they used. Yet, the shared-bottom multi-tower strucure is pretty standard in mult-tasking modeling [2, 22]. Also, none of those works are based on real time online learning. Their techniques can not fully address the challenges we talk about earlier for building an online multi-task learning framework for real time online advertising.

To the best of our knowledge, this is the first paper talking about how to successfully build an online large-scale multi-task learning framework for real time feed ads auction models to reach the launch bar.

3 DESCRIPTION OF THE FEED ADS AUCTION

3.1 Auction model predictions

Ads auction, or ads ranking, is the step where a set of ads candidates are ranked against each based on the ad's quality and the bid. Before we run an auction to rank those ads candidates, We will first generate some quality prediction scores. Different ads system may need to generate different quality scores. In this paper ,we mainly generate the following prediction scores

$$\begin{aligned} pCTR &= \mathbb{P}(y_{click} = 1|view) \\ pDismiss &= \mathbb{P}(y_{dismiss} = 1|view) \\ pClickQuality &= \mathbb{P}(y_{clickQuality} = 1|click) \end{aligned} \tag{2}$$

Here, pCTR denotes the probability that an viewed ad is clicked by the user, a.k.a, the predicted click through rate. pDismiss stands for the probability that the ad is not relevant and dismissed by the user, a.k.a, the predicted dismiss rate. And, pClickQuality is the probability the ad click leads to a good landing page experience based on post-click signals, a.k.a, the predicted good click quality rate. All labels here are binary labels. Originally, we have three independent models pCTR, pDismiss and pClickQuality models to predict the three quality scores using corresponding labels and set of ads events. Figure 2 shows the architecture of the original prediction model which uses an architecture similar to the one used

in the wider & deeper model [4]. The model first converts each raw feature (such as user feature, and ads content features) to a numeric embedding feature through a feature embedding lookup layer. Those embedding features are concatenated together and feed into a deep cross network [18] to get higher order feature interaction. The output of the deep cross network is fed into a DNN tower which consist of a stack of nonlinear activation layers. The output of each tower is used to make a binary prediction of each task. We capture the position bias by adding the position embedding feature into the logit of the each tower similar to [22]. Each model is updated via propagating the gradients from its corresponding sigmoid cross-entropy loss.

3.2 Online training platform

Figure 3 graphically illustrates our online training system which is based on the work of [13]. Our models are initialized and start training at a fixed date in the past. The Models advance quickly by training over historical data. Once the model has consumed the historical data, it catches up to the real time and trains on new events as they arrive. The model evaluates on each data point as it arrives and before it is trained by the model. The training data we use are streaming logs data of the feed ads interaction which consists of millions of training events per day. Unlike traditional offline machine learning model training, the model never fully converges, because it is only allowed to see each training event once. We do not have a fixed validation/test set. Instead, when a model checkpoint is trained base on data up to time t, it is evaluated on the log data from the time range $[t, t + \delta_t]$ which have not been seen by the model yet. In the online training framework, the model is able to keep updated with the most recent changes happening in the world, and thus ensure the model's freshness which is critical for a real advertising platform. In this paper, our goal is to build a multi-task learning framework so that we only have one model to make three predictions in one shot. The multi-task model is still trained on this online training platform and should perform not worse or even better than the original individual model.

3.3 Ads Ranking by Auction

Once we get the ads quality prediction scores, We rank the ad candidates by a ranking formulation like the following

$$\begin{aligned} AuctionScore &= f(bid, pCTR, pDismiss, pClickQuality) \\ &= bid * pCTR + QualityThreshold \end{aligned} \tag{3}$$

where $QualityThreshold$ is a function of pCTR, pDismiss and pClickQuality

$$QualityThreshold = g(pCTR, pDismiss, pClickQuality) \tag{4}$$

QualityThreshold term is used to guarantee that each ad candidate needs to meet some quality bar beside bid and clickrate, if that ad wants to be competitive. For example, higher pClickQuality and lower pDismiss will increase the auction score. The details of the function of the auction score are out of scope of this paper. Each feed ad candidate will get an auction score computed by Eq. 3 and is ranked against each other.

4 ONLINE MULTI-TASK LEARNING FRAMEWORK FOR FEED ADS AUCTION

Our goal is to build a multi-task framework so that we only have one model to make three predictions in one shot. The multi-task model is still trained on this online training platform and should perform no worse or even better than the original individual model. To achieve this goal, there are a lot of challenges we need to conquer.

4.1 Handle different label delay time for different tasks

During online training, each task's positive label is available for training at different time. The label delay is the time period between the moment an ad is seen by the use (an impression) and the moment when a corresponding user action label finally get logged and available for training. Due to various reasons (e.g., delays in label feedback and the engineering pipelines for getting the signals), the label delay time of different tasks can be very different. For our cases discussed in this paper, click action usually has only short delay while the dismiss and post-click action have much longer delay. That means the click label will usually be available within a short period of time after the ad is shown, while most dismiss and post-click labels (used to measure click quality) will not be available quickly. Because of the label delay, the model will need to wait sometime in order to get matured labels before being training on the events. We call this waiting time as training delay. When each model is trained separately, pCTR model's training delay time is h_2 hours. On the other hand, pDismiss and pClickQuality models have h_1 hours' training delay where $h_1 > 8 * h_2$.

However, the different label delays of different tasks introduce new challenges when we try to consolidate all the predictions tasks into a single model on the online training platform. Remember, in online training, once the model has consumed the historical data, it catches up to the present time and is trained on real time events as they arrive. If we set the training delay time as long as h_1 for all task, the pCTR task will not be able to catch up with the latest changes. On the other hand, if the delay time is set as short as h_2 hours for all heads, the pDismiss and pClickQuality tasks will not be able to get most positive labels once the model catchs up to real time training data. As a result, model's performance will deteriorate significantly because our pDismiss and pClickQuality tasks are only trained on the negative labels afterward.

A recent work in [1] discussed how they attempt to address the conversion delay issue in a single conversion prediction task by predicting immature label using a model trained with matured label. Differing from the conversion prediction where the delay range can be very wide (up to months) in a single task, we need to handle different delays in different tasks but delay in each task has a much narrower range. We use a similar but different approach to solve label delay time issue in the multi-task modeling. We propose the multi-stage training to allow different tasks to be trained on events with different training delay under the multi-task learning framework. Figure 4 graphically demonstrates how the multi-stage training works to handle different label delays of different tasks in a multi-task learning framework. The training is split into two stages and each stage will have an associated model checkpoint. In the first stage, all the training events' age is over h_1 hours. Because the

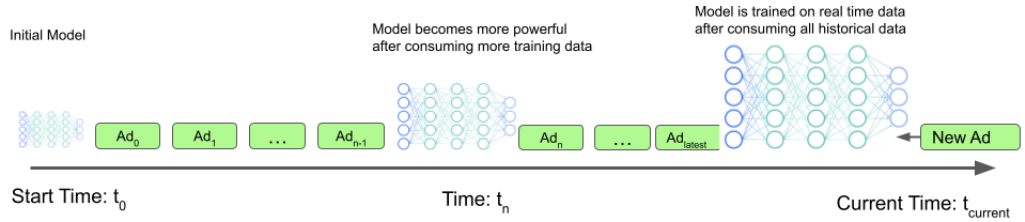


Figure 3: Online training framework

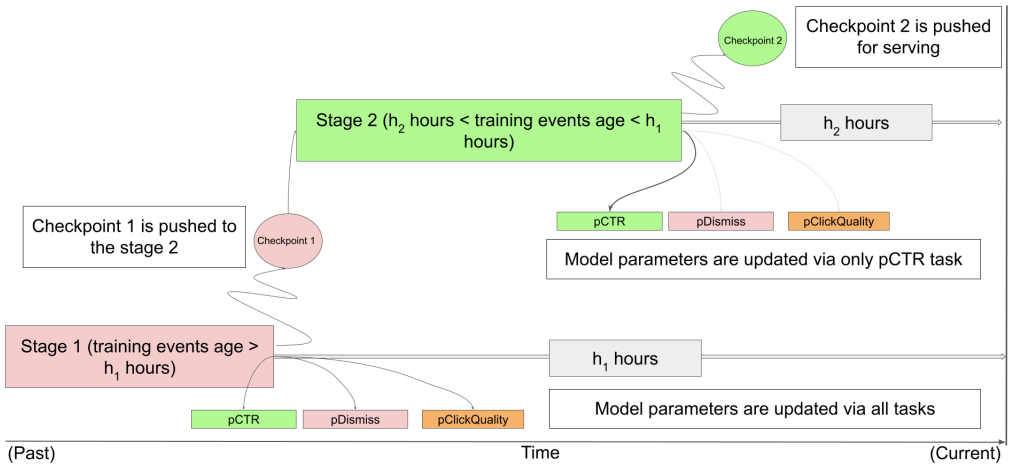


Figure 4: Multistage Training to Handle Different Label Delays

labels are mature for all tasks in stage 1, we allow all tasks to update the multi-head model at the same time. Once the model catches up to the latest events in the stage 1 (the latest events in stage 1 is h_1 hours old), a checkpoint will be generated and pushed to the stage 2. In stage 2, all training events' age is between h_1 hours and h_2 hours. In this stage, the click label for pCTR task is mature but dismiss and post-click labels are not for pDismiss and pClickQuality tasks. So, we only allow the pCTR task to update the multi-head model. Once the model in stage 2 catches up to the latest data (the latest event in stage 2 is h_2 hours old), the checkpoint will be pushed for serving. To restrict which task is allowed to update the multi-task model in each stage, we specify an age feature for each training event and modify each task's loss via the following equations

$$Loss_{pCTR} = Loss_{pCTR} * \mathbb{1}\{age >= h_2\} \tag{5}$$

$$Loss_{pDismiss} = Loss_{pDismiss} * \mathbb{1}\{age >= h_1\} \tag{6}$$

$$Loss_{pClickQuality} = Loss_{pClickQuality} * \mathbb{1}\{age >= h_1\} \tag{7}$$

It is important to emphasize that only using the restricted loss with a single training stage will not work in the online training framework. When the model catches up to the latest data, the age of the training events are only h_2 hours, and thus will be filtered out by the pDismiss and pClickQuality tasks, resulting in no training data being used by the pDismiss and pClickQuality tasks.

By using the multi-stage training and event age restricted loss, we allow all tasks to update the multi-task model in stage 1 where all tasks' labels are mature, but only allow the pCTR task to update the mobel in stage 2 where only the click label is matured. In this way, we can guarantee that all the tasks are trained on the their own most recent mature labels.

4.2 Modeling Task Interaction with Multi-gate Mixture-of-Experts

We start with out multi-task model using a shared-bottom model structure [2, 22], as shown in Figure 5 (A). The three tasks share the embedding feature layers similar to the embedding in the Wider & Deeper [4] and the deep cross network (DCN) layer [18, 19]. Same to the DNN tower in individual task, each task is associated with a tower consisting of a stack of nonlinear activation layers.

To model the task interaction better, we extend the shared-bottom structure to Multi-gate Mixture-of-Experts (MMoE) [30] as shown in Figure 5 (B) which has been proved to be very effective in modeling multi-task interaction in real Youtube recommendation system. In MMoE, we have a set of experts shared by different tasks. Each expert is stack of ReLu layers and learn a specific semantic

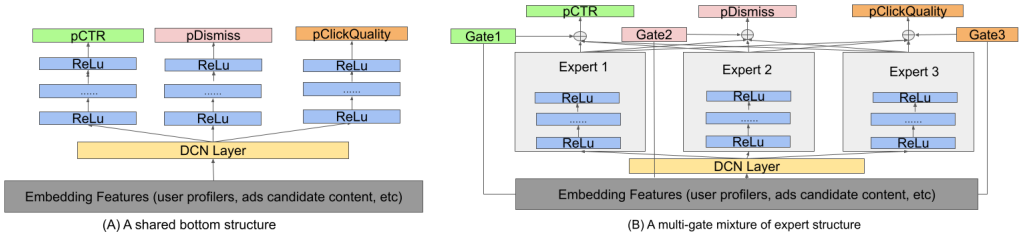


Figure 5: The architecture of shared bottom and multi-gated mixture-of-experts (MMoE)

information of the input features. Each task decides how much information it wants to take from each expert via a gating network. The gate takes the embedding layer's output as the input and outputs a vector of weights. The input to a task prediction is a weighted sum of the outputs of all experts weighted by the weights from a specific gate. Unlike some other interaction techniques mentioned earlier, MMoE is very easy to scale out by just adding more gates for more tasks and/or more experts for larger model capacity.

Mathematically, let's denote x as the input embedding features, $gate^k(x)$ as the output of gate k, and $expert_i(x)$ as the output of expert i. The gating networks are linear transformation of input with a softmax layer. The output of the gate for task k is

$$gate^k(x) = \text{softmax}(W_{g^k}(x)) \tag{8}$$

where W_{g^k} are the learning parameters for gate k. So, the input of task k is

$$task^k(x) = \sum_{i=1}^{n} gate_i^k(x) * expert_i(x) \tag{9}$$

The binary prediction of task k is

$$p^k(x) = \text{sigmoid}(h_k(task^k(x)) + pos_k) \tag{10}$$

where h_k a hidden layer with output dimension 1 and pos_k stands for the position information.

Hyper-parameters of MMoE includes but not limited to the depth and width of each expert, the number of experts, the position of the gates, and the input to the gates. The hard restriction is that the number of gates must be equal to the number of tasks and the dimension of the gate output must be equal to the number of the experts. This makes MMoE very easy to scale out by adding more gates for more task, or more experts for larger model capacity. In our experiment, we choose the depth and width of each expert to be equal to the ones of each tower in the shared-bottom model. The number of experts is 3. This ensures that the model size of the MMoE structure is comparable to the shared-bottom structure except the negligible parameters introduced by three gates.

4.3 Assigning Different training events to different tasks

When tasks are trained separately, each of them may use different set of training events. For example, $pCTR = p(click|view)$ predicts the click-through-rate of a viewed ad. On the other hand, $pClickQuality = p(clickQuality|click)$ predicts the probability that the click will leads to a good landing page experience, and thus the prediction is based on a clicked ad. When each task is trained separately, we can easily assign different training events to different tasks.

However, in the multi-task training framework, without applying additional techniques, all the tasks will have to use the same training events as all tasks share the bottom embedding layers. Assume we only use the viewed events as the training data to train all tasks, we are actually predicting $p(clickQuality|view)$ which will be significantly smaller than $p(clickQuality|click)$. To solve this issue, we restrict the loss of a specific event type to a corresponding task. Mathematically, it can be expressed as the following

$$Loss_{pCTR} = Loss_{pCTR} * \mathbb{1}\{isViewed\} \tag{11}$$

$$Loss_{pDismiss} = Loss_{pDismiss} * \mathbb{1}\{isViewed\} \tag{12}$$

$$Loss_{pClickQuality} = Loss_{pClickQuality} * \mathbb{1}\{isClicked\} \tag{13}$$

For the pClickQuality prediction task, we only allow the loss from clicked training event to propagate through the network. For the pCTR and pDismiss prediction tasks, we only use the loss from the events which have been viewed by the users. Graphically, it is equally to apply event filters on different tasks as shown in the Figure 6 such that each task is assigned with different training events.

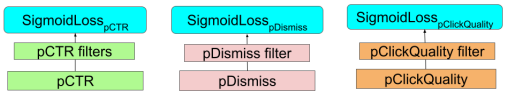


Figure 6: Use Different filters to assign different training events to different tasks.

4.4 Learning loss weights of different tasks

A common and practical approach to combining multi objective losses is to perform a weighted sum of the losses of different tasks. The loss weights serve like a tuning knobs to tune the importance of a specific task among all other tasks. Increasing the weight of one task usually leads to a bit better performance of this task in the cost of other tasks' performance.

However, determining the loss weights for different tasks in the multi-task learning is still an active and challenging research topic. There is no one-fit-all solution of tuning the loss weights of

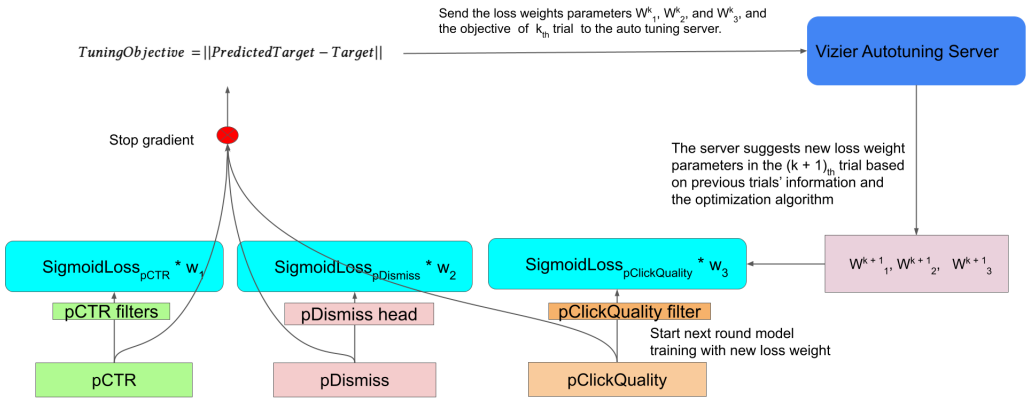


Figure 7: System to automatically learn the loss weight of different tasks

different tasks. However, as mentioned in Section 2, most existing loss weight strategies can not be easily generalized to the online advertising problem and is not very practical here. In practice, multi-task models still manually tune task loss weights and make the decision based on the trade-off between different tasks' offline and online metrics. However, the manual tuning is very tedious, not optimal, neither scalable for adding more tasks in the future. In this paper, we would like to demonstrate a novel and more systematic approach to tune the loss weights of different ads auction prediction tasks.

Instead of solving it as multi-objective optimization, we define a single multi-task learning objective which can guide the task loss weight tuning. Inspired by auction score in Eq. 3, we define the predict target as

$$\begin{aligned} PredictedTarget &= bid * pCTR \\ &+ g(pCTR, pDismiss, pClickQuality) \end{aligned} \tag{14}$$

The ground truth for $pCTR$, $pClickQuality$, and $pDismiss$, are $isClick$, $clickQuality$ and $isDismiss$, respectively. bid is logged in the system and can be directed used. So, we have the ground truth target for the predicted target

$$Target = bid * isClick + g(isClick, isDismiss, clickQuality) \tag{15}$$

We define a single multi-task learning objective as the one shown in Eq. 16 which can be used to guide the task loss weight tuning

$$TuningObjective = ||PredictedTarget - Target|| \tag{16}$$

After we have defined the tuning objective, the remaining work is to start task weight parameter tuning via some auto-parameter tuning servers like Vizier [7] so that the multi-task objective will be minimized. Figure 7 illustrates the end-to-end process of how we tune the loss weights. In each trial, the multi-task model is trained with a fixed set of weights W_1, W_2, W_3 using a fixed range of recent data. Please note that we do not propagate the gradient from this multi-task tuning objective. To mimic the auction during serving time, the tuning objective was computed on non-filtered events and is only used to guide the loss weight tuning. The multi-task model is still updated via the gradient from each task's sigmoid loss. After the model finished training on the data set, the corresponding multi-task objective is computed and recorded. Then, the current loss weights and the multi-task objective are send to the auto-tuning service. Based on the previous trials' information and the optimization algorithm, the server will suggest new loss weights we should try in next trial to reduce the multi-task objective further. We can run multiple trials in parallel at the same time. After running multiple trials offline, the server will suggest the best loss weights which will be used in our real time multi-task model trained on the online platform.

We demonstrate that this method can effectively and efficiently find a better sweet spot of the loss weights and improve the multi-task's model performance incrementally. We also tried only using the multi-task objective to train the multi-head model which, however, lead to drastic model behavior changes. Future works could try propagating a small amount of gradient from the multi-task objective as complement to existing sigmoid loss.

5 EXPERIMENTAL RESULT

We have gone through multiple launches by applying the techniques we discussed here. We first launched the baseline multi-task model based on a shared bottom deep neural network structure using multi-stage training to handle different label delays. We manually tuned a fixed set of weights, looked at each individual task' offline metric and chose a loss weight based on some heuristic rules. In the second launch, we applied the Multi-gate Mixture-of-Experts (MMoE) to learn the task interaction better. In the third launch, we used a novel auto-parameter tuning algorithm to automatically and systematically find the loss weights of different tasks. This lets us be able to measure the impact of each component incrementally.

Our primary offline metric is AUCLoss which is used to assess the binary prediction accuracy. The lower the AUCLoss the better the model performance. In our online learning platform, we do not have a fixed validation/test set. Instead, at any time t, the

model checkpoint will be evaluated on the data from the time range $[t, t + \delta_t]$ which have not been seen by the model. The loss will be averaged cross the evaluated data. We mainly focus on the average AUCLoss after the model starts consuming live data. Regarding the online metric, we only list a few related metrics such click through rate, dismiss rate (lower the better), good click quality rate, resource and latency. Although in reality, we use more metrics to assess the quality of our models. For example, despite revenue being a very important metric, we do not explicitly discuss the revenue impact here even with all positive impact. The statistically significant results are highlighted with green for positive impact and red for negative impact.

5.1 Baseline multi-task model

In the baseline multi-task model, we applied all the techniques we mentioned in Section 4 except we used a shared-bottom network and just manually tuned the task loss weights without using the proposed loss weight optimization algorithm. The goal here is to make sure we can get a solid baseline by first addressing the fundamental multitask challenging.

To verify the the label delay issues can be addressed by using multi-stage training with restricted loss. We train two set of models one of which use the delay handing techniques and the other does not. In Figure 8, We can see that the models do not have quality degradation if the label delay issues are handled appropriately.

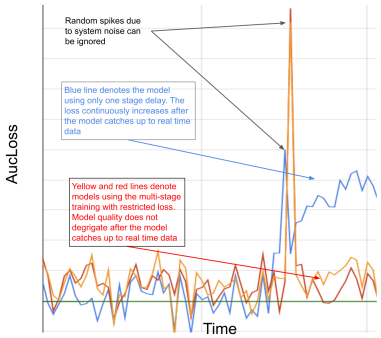


Figure 8: Offline metrics of the models using the delay handling techniques V.S. the one not using it

We got solid offline improvement. The AUCLoss of the pCTR task and pDismiss tasks were reduced by more than 0.6% and 8%, respectively. It shows that the pDismiss, which has very sparse labels, can be largely benefited from the multi-task learning via transfer learning. The Table 1 shows the offline results of our multi-task model compared to the production model. We also saw positive online metric improvement. We increased the click-through-rate CTR by +1.3% and decreased the dismiss rate by 20%, and significantly reduced our total model size by 63.2% (majority model size came from the embedding table and DCN layer) and server cpu usage by 16% (1 v.s 3 model inference RPC calls). The latency is neutral because of good parallelism and same model depth. The engineering velocity brought by multi-tasking is also substantial because any new idea can be tried on three tasks simultaneously and one launch could potentially translate to three original launches.

Table 1: The relative AUC Loss of the baseline multi-task models compared to the production models

	pCTR	pDismiss	pClickQuality
Multitask Basline	-0.6%	-8.0%	neutral

Table 2: The online metric of the baseline multi-task models compared to the production models

CTR	+1.3%
DismissRate	-20%
GoodClickQualityRate	neutral
Server CPU Usage	-16.5%
Model Training & Serving Resource	-63.2%
Inference Latency	neutral

Table 3: The relative AUC loss of the MMoE model compared to the multi-head baseline model

	pCTR	pDismiss	pClickQuality
MMoE	-0.3%	neutral	neutral

Table 4: The online metric of the MMoE model compared to the multi-head baseline model

CTR	+1.41%
DismissRate	neutral
GoodClickQualityRate	neutral
Server CPU Usage	neutral
Model Training & Serving Resource	neutral
Inference Latency	neutral

5.2 Multi-gate Mixture-of-Experts

By adopting the MMoE structure, we got additional gain on top of the baseline multi-task model launch in Section 5.1. The AUC Loss of the pCTR task was reduced by 0.3% and the click-through-rate CTR was increased by +1.41% as shown in Table 4. Improving CTR is important, since it's a dominant signal in the long term value function used in the ads auction. By learning the interaction better via MMoE, we can continue to improve the model performance.

5.3 Objective weights Optimization

After two successful launches in Section 5.1 and 5.2, we used two optimization algrithms 'grid search' and 'bayesian optimization' with the single multi-task tuning objective defined in Section 4.4. We also specified the same search space of three loss weights and same number of trials for both algorithms. Once the best weights (giving smallest turning objective defined in Eq. 16) were suggested by the vizier, we will use them to train full online multi-task models.

Table 5 shows the offline metrics of the models using best weights with different optimization algorithms. The multi-task tuning objective, as defined in Eq. 16, of the grid search was better than the bayesian optimization by 0.3%. We can also see that the offline metric of the grid search had almost uniform quality improvement

Table 5: The relative AUC loss of the multi-task models with different weight optimization algorithms compared to the MMoE model.

algorithm	pCTR	pDismiss	pClickQuality
Bayesian Optimization	neutral	+1.58%	-1.12%
Grid Search	neutral	-0.68%	-0.7%

Table 6: The online metric of the multi-task model with grid search optimization compared to the MMoE model.

CTR	neutral
DismissRate	-2.1%
GoodClickQualityRate	+0.2%
Server CPU Usage	neutral
Model Training & Serving Resource	neutral
Inference Latency	neutral

on all tasks, while the bayesian optimization hurted the pDismiss task despite of having better performance on pClickQuality task compared to grid search. So, we decided to use the loss weights suggested by the the grid search algorithm for online experiment because it provided a better sweet spot for all tasks. The reason 'grid search' performed a bit better was likely because it covered the whole search space we specified while the 'bayesian optimizton' might get stuck in some local optimal too early.We conducted the online experiment with the model using the loss weight suggested by the grid search. We saw online metric improvement as shown in Table 6. This demonstrates that using the multi-task objective which is derived from the auction formulation can guide us to find a better optimal loss weights of different task.

6 CONCLUSION AND FUTURE WORK

We proposed a large scale online multi-task learning framework for feed ads auction and addressed the fundamental changes unique to this online multi-tasking framework for online advertising with novel and practical techniques. These techniques leads to a series of successful launches on a real industry-scale feed ads platform. This multi-task framework can also be scaled to take more ads auction tasks such as conversion prediction task using post-click data. This paves the way to build a practical online multi-task learning framework for online advertising. Some other novel research ideas could also be explored as future work to incrementally improve the task interaction [20] and mitigate task conflict [21].

REFERENCES

[1] Ashwinkumar Badanidiyuru, Andrew Evdokimov, Vinodh Krishnan, Pan Li, Wynn Vonnegut, and Jayden Wang. 2021. Handling many conversions per click in modeling delayed. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore). ACM, 9 pages. http://papers.adkdd.org/2021/papers/adkdd21-badanidiyuru-handling.pdf

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

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *Proceedings of the 35th International Conference on Machine Learning* (Stockholm, Sweden).

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, and Mustafa Ispir. 2016. Wide and deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender system* (Boston, MA, USA). ACM. https://doi.org/10.1145/2988450.2988454

[5] 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* (Helsinki, Finland). ACM, 160–167. https://doi.org/10.1145/1390156.1390177

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston MA, USA). IEEE. https://doi.org/10.1145/2959100.2959190

[7] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. 2017. Google Vizier: A Service for Black-Box Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada). ACM, 1487–1495. https://doi.org/10.1145/3097983.3098043

[8] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA). IEEE. https://doi.org/10.1109/CVPR.2018.00781

[9] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qingfu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Vancouver Canada). ACM, 12060–12070. https://doi.org/10.5555/3454287.3455367

[10] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S. Yu. 2017. Learning multiple tasks with multilinear relationship networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, CA, USA). ACM, 1593—-1602. https://doi.org/10.5555/3294771.3294923

[11] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 2020. 12-in-1: Multi-Task Vision and Language Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA, USA). IEEE. https://doi.org/10.1109/CVPR42600.2020.01045

[12] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed 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* (London, United Kingdom). ACM, 1930–1939. https://doi.org/10.1145/3219819.3220007

[13] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (Chicago, IL, USA). IEEE. https://doi.org/10.1145/2487575.2488200

[14] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada). ACM, 525–536. https://doi.org/10.5555/3326943.3326992

[15] Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which Tasks Should Be Learned Together in Multi-task Learning?. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*.

[16] Simon Vandenhende1, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. 2020. Branched Multi-Task Networks:Deciding What Layers To Share. In *The 31st British Machine Vision Virtual Conference*.

[17] Ernest Wang. 2020. *How we use AutoML, Multi-task learning and Multi-tower models for Pinterest Ads.* https://medium.com/pinterest-engineering/how-we-use-automl-multi-task-learning-and-multi-tower-models-for-pinterest-ads-db966c3dc99e

[18] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada). ACM, 1487–1495. https://doi.org/10.1145/3124749.3124754

[19] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia). ACM, 1785–1797. https://doi.org/10.1145/3442381.3450078

[20] Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, and Ed H. Chi. 2020. Small Towers Make Big Differences. https://arxiv.org/pdf/2008.05808.pdf

[21] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, and Sergey Levine. 2020. Gradient Surgery for Multi-Task Learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Virtual Conference).

[22] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark). ACM, 43–51. https://doi.org/10.1145/3298689.3346997