FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction

Kelong Mao[1*], Jieming Zhu[2*], Liangcai Su[3], Guohao Cai[2], Yuru Li[2], Zhenhua Dong[2]

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Huawei Noah's Ark Lab
[3]Tsinghua University
kyriemkl@gmail.com, jiemingzhu@ieee.org, sulc21@mails.tsinghua.edu.cn

Abstract

Click-through rate (CTR) prediction is one of the fundamental tasks for online advertising and recommendation. While multi-layer perceptron (MLP) serves as a core component in many deep CTR prediction models, it has been widely recognized that applying a vanilla MLP network alone is inefficient in learning multiplicative feature interactions. As such, many two-stream interaction models (e.g., DeepFM and DCN) have been proposed by integrating an MLP network with another dedicated network for enhanced CTR prediction. As the MLP stream learns feature interactions implicitly, existing research focuses mainly on enhancing explicit feature interactions in the complementary stream. In contrast, our empirical study shows that a well-tuned two-stream MLP model that simply combines two MLPs can even achieve surprisingly good performance, which has never been reported before by existing work. Based on this observation, we further propose feature selection and interaction aggregation layers that can be easily plugged to make an enhanced two-stream MLP model, FinalMLP. In this way, it not only enables differentiated feature inputs but also effectively fuses stream-level interactions across two streams. Our evaluation results on four open benchmark datasets as well as an online A/B test in our industrial system show that FinalMLP achieves better performance than many sophisticated two-stream CTR models. Our source code will be available at MindSpore/models and FuxiCTR/model_zoo.

Introduction

Click-through rate (CTR) prediction is a fundamental task in online advertising and recommender systems (Cheng et al. 2016; He et al. 2014). The accuracy of CTR prediction not only has a direct effect on user engagement but also significantly influences the revenue of business providers. One of the key challenges in CTR prediction is to learn complex relationships among features such that a model can still generalize well in case of rare feature interactions. Multi-layer perceptron (MLP), as a powerful and versatile component in deep learning, has become a core building block of various CTR prediction models (Zhu et al. 2021). Although MLP is known to be a universal approximator in theory, it has been

widely recognized that in practice applying a vanilla MLP network is inefficient to learn multiplicative feature interactions (e.g., dot) (Wang et al. 2017, 2021; Rendle et al. 2020).

To enhance the capability of learning explicit feature interactions (2nd- or 3rd-order features), a variety of feature interaction networks have been proposed. Typical examples include factorization machines (FM) (Rendle 2010), cross network (Wang et al. 2017), compressed interaction network (CIN) (Lian et al. 2018), self-attention based interaction (Song et al. 2019), adaptive factorization network (AFN) (Cheng, Shen, and Huang 2020), and so on. These networks introduce inductive bias for learning feature interactions efficiently but lose the expressiveness of MLP as our experiments shown in Table 3. As such, two-stream CTR prediction models have been widely employed, such as DeepFM (Guo et al. 2017), DCN (Wang et al. 2017), xDeepFM (Lian et al. 2018), and AutoInt+ (Song et al. 2019), which integrate both an MLP network and a dedicated feature interaction network together for enhanced CTR prediction. Concretely, the MLP stream learns feature interactions implicitly, while the other stream enhances explicit feature interactions in a complementary way. Due to their effectiveness, two-stream models have become a popular choice for industrial deployment (Zhang et al. 2021).

Although many existing studies have validated the effectiveness of two-stream models against a single MLP model, none of them reports a performance comparison to a two-stream MLP model that simply combines two MLP networks in parallel (denoted as DualMLP). Therefore, our work makes the first effort to characterize the performance of DualMLP. Our empirical study on open benchmark datasets shows that DualMLP, despite its simplicity, can achieve surprisingly good performance, which is comparable to or even better than many well-designed two-stream models (see our experiments). This observation motivates us to study the potential of such a two-stream MLP model and further extend it to build a simple yet strong model for CTR prediction.

Two-stream models in fact can be viewed as an ensemble of two parallel networks. One advantage of these two-stream models is that each stream can learn feature interactions from a different perspective and thus complements each other to achieve better performance. For instance, Wide&Deep (Cheng et al. 2016) and DeepFM (Guo

---

[*]Both authors contributed equally. Jieming Zhu is the corresponding author.

et al. 2017) propose to use one stream to capture low-order feature interactions and another to learn high-order feature interactions. DCN (Wang et al. 2017) and AutoInt+ (Song et al. 2019) advocate learning explicit feature interactions and implicit feature interactions in two streams respectively. xDeepFM (Lian et al. 2018) further enhances feature interaction learning from vector-wise and bit-wise perspectives. These previous results verify that the differentiation (or diversity) of two network streams makes a big impact on the effectiveness of two-stream models.

Compared to the existing two-stream models that resort to designing different network structures (e.g., CrossNet (Wang et al. 2017) and CIN (Lian et al. 2018)) to enable stream differentiation, DualMLP is limited in that both streams are simply MLP networks. Our preliminary experiments also reveal that DualMLP, when tuned with different network sizes (w.r.t., number of layers or units) for two MLPs, can achieve better performance. This result promotes us to further explore how to enlarge the differentiation of two streams to improve DualMLP as a base model. In addition, existing two-stream models often combine two streams via summation or concatenation, which may waste the opportunity to model the high-level (i.e., stream-level) feature interactions. How to better fuse the stream outputs becomes another research problem that deserves further exploration.

To address these problems, in this paper, we build an enhanced two-stream MLP model, namely FinalMLP, which integrates **f**eature selection and **in**teraction **a**ggregation **l**ayers on top of two **MLP** module networks. More specifically, we propose a stream-specific feature selection layer that allows obtaining gating-based feature importance weights for soft feature selection. That is, the feature gating can be computed from different views via conditioning on learnable parameters, user features, or item features, which produces global, user-specific, or item-specific feature importance weights respectively. By flexibly choosing different gating-condition features, we are able to derive stream-specific features for each stream and thus enhance the differentiation of feature inputs for complementary feature interaction learning of two streams. To fuse the stream outputs with stream-level feature interaction, we propose an interaction aggregation layer based on second-order bilinear fusion (Lin, RoyChowdhury, and Maji 2015; Li et al. 2017). To reduce the computational complexity, we further decompose the computation into k sub-groups, which leads to efficient multi-head bilinear fusion. Both feature selection and interaction aggregation layers can be easily plugged into existing two-stream models.

Our experimental results on four open benchmark datasets show that FinalMLP outperforms the existing two-stream models and attains new state-of-the-art performance. Furthermore, we validate its effectiveness in industrial settings through both offline evaluation and online A/B testing, where FinalMLP also shows significant performance improvement over the deployed baseline. We envision that the simple yet effective FinalMLP model could serve as a new strong baseline for future developments of two-stream CTR models. The main contributions of this paper are summarized as follows:

- To our knowledge, this is the first work that empirically demonstrates the surprising effectiveness of a two-stream MLP model, which may be contrary to popular belief in the literature.
- We propose FinalMLP, an enhanced two-stream MLP model with pluggable feature selection and interaction aggregation layers.
- Both Offline experiments on benchmark datasets and an online A/B test in production systems have been conducted to validate the effectiveness of FinalMLP.

Background and Related Work

In this section, we briefly review the framework and representative two-stream models for CTR prediction.

Framework of Two-Stream CTR Models

We illustrate a framework of two-stream CTR models in Figure 1(b), which consists of the following key components.

Feature Embedding Embedding is a common way to map high-dimensional and sparse raw features into dense numeric representations. Specifically, suppose that the raw input feature is $x = \{x_1, ..., x_M\}$ with M feature fields, where x_i is the feature of the i-th field. In general, x_i can be a *categorical*, *multi-valued*, or *numerical* feature. Each of them can be transformed into embedding vectors accordingly. Interested readers may refer to (Zhu et al. 2021) for more details on feature embedding methods. Then, these feature embeddings will be concatenated and fed into the following layer.

Feature Selection Feature selection is an *optional* layer in the framework of two-stream CTR models. In practice, feature selection is usually performed through offline statistical analysis or model training with difference comparison (Pechuán, Ponce, and de Lourdes Martínez-Villaseñor 2016). Instead of hard feature selection, in this work, we focus on the soft feature selection through the feature gating mechanism (Huang, Zhang, and Zhang 2019; Guan et al. 2021), which aims to obtain feature importance weights to help amplify important features while suppressing noisy features. In this work, we study stream-specific feature selection to enable differentiated stream inputs.

Two-Stream Feature Interaction The key feature of two-stream CTR models is to employ two parallel networks to learn feature interactions from different views. Basically, each stream can adopt any type of feature interaction network (e.g., FM (Rendle 2010), CrossNet (Wang et al. 2017), and MLP). Existing work typically applies two different network structures to the two streams in order to learn complementary feature interactions (e.g., explicit vs. implicit, bitwise vs. vector-wise). In this work, we make the first attempt to use two MLP networks as two streams.

Stream-Level Fusion Stream-level fusion is required to fuse the outputs of two streams to obtain the final predicted click probability $\hat{y}$. Suppose $\mathbf{o_1}$ and $\mathbf{o_2}$ as two output representations, it can be formulated as: $\hat{y} = \sigma(w^T \mathcal{F}(\mathbf{o_1}, \mathbf{o_2}))$, where $\mathcal{F}$ denotes the fusion operation which is commonly
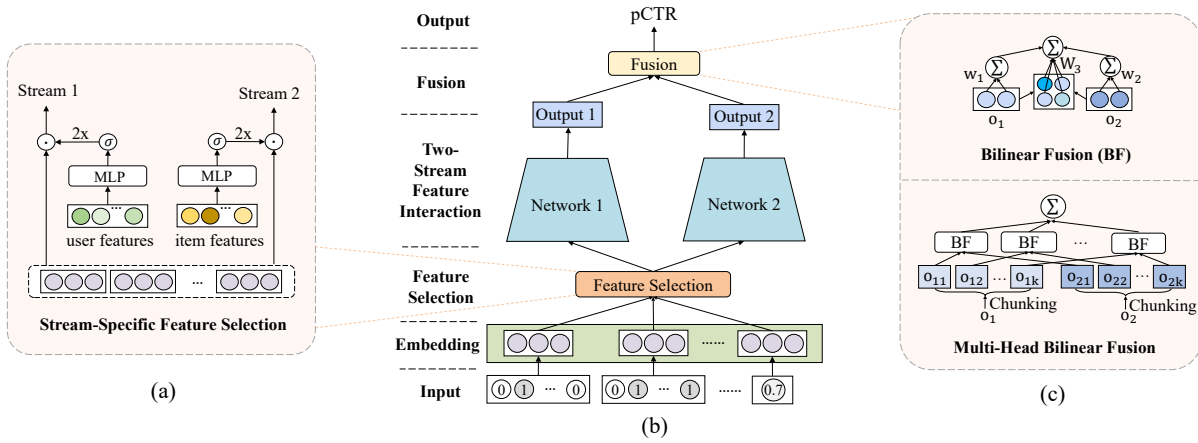
Figure 1: (a) An illustration of stream-specific feature selection. (b) A general framework of two-stream CTR models. (c) The multi-head bilinear fusion.

set as summation or concatenation[1]. w denotes a linear function to map the output dimension to 1 when necessary. σ is the sigmoid function. Existing work only performs a first-order linear combination of stream outputs, so it fails to mine the stream-level feature interactions. In this work, we explore a second-order bilinear function fo stream-level interaction aggregation.

Representative Two-Stream CTR Models

We summarize some representative two-stream models that cover a wide spectrum of studies on CTR prediction.

- **Wide&Deep**: Wide&Deep (Cheng et al. 2016) is a classical two-stream feature interaction learning framework that combines a generalized linear model (wide stream) and an MLP network (deep stream).
- **DeepFM**: DeepFM (Guo et al. 2017) extends Wide&Deep by replacing the wide stream with FM to learn second-order feature interactions explicitly.
- **DCN**: In DCN (Wang et al. 2017), a cross network is proposed as one stream to perform high-order feature interactions in an explicit way, while another MLP stream learns feature interactions implicitly.
- **xDeepFM**: xDeepFM (Lian et al. 2018) employs a compressed interaction network (CIN) to capture high-order feature interactions in a vector-wise way, and also adopts MLP as another stream to learn bit-wise feature interactions.
- **AutoInt+**: AutoInt (Song et al. 2019) applies self-attention networks to learning high-order feature interactions. AutoInt+ integrates AutoInt and MLP as two complementary streams.
- **AFN+**: AFN (Cheng, Shen, and Huang 2020) leverages logarithmic transformation layers to learn adaptive-order feature interactions. AFN+ combines AFN with MLP in a two-stream manner.

---

[1]In case that the output dimension is 1, the fusion operation of concatenation is approximately equivalent to summation because $w^T[o_1, o_2] = [w_1, w_2]^T[o_1, o_2] = w_1^T o_1 + w_2^T o_2$.

- **DeepIM**: In DeepIM (Yu et al. 2020), an interaction machine (IM) module is proposed to efficiently compute high-order feature interactions. It uses IM and MLP separately in two streams.
- **MaskNet**: In MaskNet (Wang, She, and Zhang 2021), a MaskBlock is proposed by combining layer normalization, instance-guided mask, and feed-forward layer. The parallel MaskNet is a two-stream model that uses two MaskBlocks in parallel.
- **DCN-V2**: DCN-V2 (Wang et al. 2021) improves DCN with a more expressive cross network to better capture explicit feature interactions. It still uses MLP as another stream in the parallel version.
- **EDCN**: EDCN (Chen et al. 2021) is not a strict two-stream model, since it proposes a bridge module and a regulation module to bridge the information fusion between hidden layers of two streams. However, its operations limit each stream to having the same size of hidden layers and units, reducing flexibility.

Our Model

In this section, we first present the simple two-stream MLP base model, **DualMLP**. Then, we describe two pluggable modules, feature selection and interaction aggregation layers, which results in our enhanced model, **FinalMLP**.

Two-Stream MLP Model

Despite its simplicity, to the best of our knowledge, the two-stream MLP model has not been reported before by previous work. Thus, we make the first effort to study such a model for CTR prediction, denoted as DualMLP, which simply combines two independent MLP networks as two streams. Specifically, the two-stream MLP model can be formulated as follows:

$$\mathbf{o_1} = MLP_1(\mathbf{h_1}), \qquad (1)$$
$$\mathbf{o_2} = MLP_2(\mathbf{h_2}), \qquad (2)$$

where MLP_1 and MLP_2 are two MLP networks. The size of both MLPs (w.r.t. hidden layers and units) can be set differently according to data. $\mathbf{h_1}$ and $\mathbf{h_2}$ denote the feature inputs while $\mathbf{o_1}$ and $\mathbf{o_2}$ are the output representations of the two streams, respectively.

In most previous work (Wang et al. 2017; Guo et al. 2017; Cheng, Shen, and Huang 2020), the feature inputs $\mathbf{h_1}$ and $\mathbf{h_2}$ are usually set as the same one, which is a concatenation of feature embeddings $\mathbf{e}$ (optionally with some pooling), i.e., $\mathbf{h_1} = \mathbf{h_2} = \mathbf{e}$. Meanwhile, the stream outputs are often fused via simple operations, such as summation and concatenation, ignoring stream-level interactions. In the following, we introduce two modules that can be plugged into the inputs and outputs respectively to enhance the two-stream MLP model.

Stream-Specific Feature Selection

Many existing studies (Guo et al. 2017; Lian et al. 2018; Wang et al. 2017; Song et al. 2019) highlight the effectiveness of combining two different feature interaction networks (e.g., implicit vs. explicit, low-order vs. high-order, bit-wise vs. vector-wise) to achieve accurate CTR prediction. Instead of designing specialized network structures, our work aims to enlarge the difference between two streams through stream-specific feature selection, which produces differentiated feature inputs.

Inspired by the gating mechanism used in MMOE (Ma et al. 2018), we propose a stream-specific feature gating module to soft-select stream-specific features, i.e., re-weighting feature inputs differently for each stream. In MMOE, gating weights are conditioned on task-specific features to re-weight expert outputs. Likewise, we perform feature gating from different views via conditioning on learnable parameters, user features, or item features, which produces global, user-specific, or item-specific feature importance weights respectively.

Specifically, we define the stream-specific feature selection layer as follows.

$$\mathbf{g_1} = Gate_1(\mathbf{x_1}), \quad \mathbf{g_2} = Gate_2(\mathbf{x_2}), \quad (3)$$
$$\mathbf{h_1} = 2\sigma(\mathbf{g_1}) \odot \mathbf{e}, \quad \mathbf{h_2} = 2\sigma(\mathbf{g_2}) \odot \mathbf{e}, \quad (4)$$

where $Gate_i$ denotes an MLP-based gating network, which takes stream-specific conditional features $\mathbf{x_i}$ as input and outputs element-wise gating weights $\mathbf{g_i}$. Note that it is flexible to either choose $\mathbf{x_i}$ from a set of user/item features or set it as learnable parameters. The feature importance weights are obtained by using the sigmoid function σ and a multiplier of 2 to transform them to the range of $[0, 2]$ with an average of 1. Given the concatenated feature embeddings $\mathbf{e}$, we can then obtain weighted feature outputs $\mathbf{h_1}$ and $\mathbf{h_2}$ via element-wise product $\odot$.

Our feature selection module allows making differentiated feature inputs for two streams by setting conditional features $\mathbf{x_i}$ from a different view. For example, Figure 1(a) demonstrates a case of user- and item-specific feature gating, which modulates each stream from the view of users and items, respectively. This reduces "homogeneous" learning between two similar MLP streams, and would enable more complementary learning of feature interactions.

Stream-Level Interaction Aggregation

Bilinear Fusion As mentioned before, existing work mostly employs summation or concatenation as the fusion layer, but these operations fail to capture stream-level feature interactions. Inspired by the widely studied bilinear pooling in the CV domain (Lin, RoyChowdhury, and Maji 2015; Li et al. 2017), we propose a bilinear interaction aggregation layer to fuse the stream outputs with stream-level feature interaction. As illustrated in Figure 1(c), the predicted click probability is formulated as follows.

$$\hat{y} = \sigma(b + \mathbf{w}_1^T\mathbf{o}_1 + \mathbf{w}_2^T\mathbf{o}_2 + \mathbf{o}_1^T\mathbf{W}_3\mathbf{o}_2), \quad (5)$$

where $b \in \mathcal{R}$, $\mathbf{w}_1 \in \mathcal{R}^{d_1 \times 1}$, $\mathbf{w}_2 \in \mathcal{R}^{d_2 \times 1}$, $\mathbf{W}_3 \in \mathcal{R}^{d_1 \times d_2}$ are learnable weights. d_1 and d_2 denote the dimensions of $\mathbf{o}_1$ and $\mathbf{o}_2$, respectively. The bilinear term $\mathbf{o}_1^T\mathbf{W_3}\mathbf{o}_2$ models the second-order interactions between $\mathbf{o}_1$ and $\mathbf{o}_2$. Especially, when $\mathbf{W}_3$ is an identity matrix, the term simulates the dot product. When setting $\mathbf{W}_3$ to a zero matrix, it degenerates to the traditional concatenation fusion with a linear layer, i.e., $b + [\mathbf{w}_1, \mathbf{w}_2]^T[\mathbf{o}_1, \mathbf{o}_2]$.

Interestingly, the bilinear fusion also has a connection to the commonly used FM model. Concretely, FM models the second-order feature interactions among a m-dimensional input feature vector $\mathbf{x}$ (via one-hot/multi-hot feature encoding and concatenation) for CTR prediction by:

$$\hat{y} = \sigma(b + \mathbf{w}^\top\mathbf{x} + \mathbf{x}^\top \mathtt{upper}(\mathbf{PP}^\top)\mathbf{x}), \quad (6)$$

where $b \in \mathcal{R}$, $\mathbf{w} \in \mathcal{R}^{m \times 1}$, $\mathbf{P} \in \mathcal{R}^{m \times d}$ are learnable weights with $d \ll m$, and $\mathtt{upper}$ selects the strictly upper triangular part of the matrix (Rendle 2010). As we can see, FM is a special form of bilinear fusion when $\mathbf{o}_1 = \mathbf{o}_2$.

However, when $\mathbf{o}_1$ and $\mathbf{o}_2$ are high-dimensional, it is parameter-intensive and computation-expensive to compute Equation (5). For example, to fuse two 1000-dimensional outputs, $\mathbf{W}_3 \in \mathcal{R}^{1000 \times 1000}$ takes up 1 million parameters and its computation becomes expensive. To reduce the computational complexity, we introduce our extended multi-head bilinear fusion in the following.

Multi-Head Bilinear Fusion Multi-head attention is appealing for the ability to combine knowledge of the same attention pooling from different representation subspaces. It leads to reduced computation and consistent performance improvements in the recent successful transformer models (Vaswani et al. 2017). Inspired by its success, we extend the bilinear fusion to a multi-head version. Specifically, instead of directly computing the bilinear term in Equation (5), we chunk each of $\mathbf{o_1}$ and $\mathbf{o_2}$ into k subspaces:

$$\mathbf{o_1} = [\mathbf{o_{11}}, ..., \mathbf{o_{1k}}], \quad (7)$$
$$\mathbf{o_2} = [\mathbf{o_{21}}, ..., \mathbf{o_{2k}}], \quad (8)$$

where k is a hyper-parameter and $\mathbf{o_{ij}}$ denotes the j-th subspace representation of the i-th output vector ($i \in \{1, 2\}$). Similar to multi-head attention, we perform the bilinear fusion in each subspace that pairs $\mathbf{o_{1j}}$ and $\mathbf{o_{2j}}$ as a group. Then, we aggregate the subspace computation by sum pooling to get the final predicted click probability:

$$\hat{y} = \sigma(\sum_{j=1}^{k} BF(\mathbf{o_{1j}}, \mathbf{o_{2j}})), \quad (9)$$

Dataset	#Instances	#Fields	#Features
Criteo	45,840,617	39	2,086,936
Avazu	40,428,967	22	1,544,250
MovieLens	2,006,859	3	90,445
Frappe	288,609	10	5,382

Table 1: The statistics of open datasets.

where BF denotes the bilinear fusion without sigmoid activation in Equation (5).

Through the subspace computation as with multi-head attention, we can theoretically reduce the number of parameters and the computation complexity of bilinear fusion by a factor of k, i.e., from $\mathcal{O}(d_1 d_2)$ to $\mathcal{O}(\frac{d_1 d_2}{k})$. Especially, when setting $k = d_1 = d_2$, it degenerates to an element-wise product fusion. If $k = 1$, it equals to the original bilinear fusion. Selecting an appropriate k realizes multi-head learning so that the model may achieve better performance. In practice, the multi-head fusions for k subspaces are computed in parallel in GPUs, which further increases efficiency.

Finally, our stream-specific feature selection and stream-level interaction aggregation modules can be plugged to produce an enhanced two-stream MLP model, FinalMLP.

Model Tranining

To train FinalMLP, we apply the widely used binary cross-entropy loss: $\mathcal{L} = -\frac{1}{N} \sum \big(y log\hat{y} + (1 - y)log(1 - \hat{y})\big)$, where y and $\hat{y}$ denote the true label and the estimated click probability respectively from each of a total of N samples.

Experiments

Experimental Setup

Datasets We experiment with four open benchmark datasets, including Criteo, Avazu, MovieLens, and Frappe. We reuse the preprocessed data by (Cheng, Shen, and Huang 2020) and follow the same settings on data splitting and preprocessing. Table 1 summaries the statistics of the datasets.

Evaluation Metric We employ AUC as one of the most widely used evaluation metrics for CTR prediction. Moreover, a 0.1-point increase in AUC is recognized as a significant improvement in CTR prediction (Cheng et al. 2016; Cheng, Shen, and Huang 2020; Wang et al. 2021).

Baselines First, we study a set of single-stream explicit feature interaction networks as follows.

- First-order: Logistic Regression (LR) (Richardson, Dominowska, and Ragno 2007).
- Second-order: FM (Rendle 2010), AFM (Xiao et al. 2017), FFM (Juan et al. 2016), FwFM (Pan et al. 2018), and FmFM (Sun et al. 2021).
- Third-order: HOFM (3rd) (Blondel et al. 2016), Cross-Net (2L) (Wang et al. 2017), CrossNetV2 (2L) (Wang et al. 2021), and CIN (2L) (Lian et al. 2018). We specially set the maximal order to "3rd" or the number of interaction layers to "2L" to obtain third-order feature interactions.

- Higher-order: CrossNet (Wang et al. 2017), CrossNetV2 (Wang et al. 2021), CIN (Lian et al. 2018), AutoInt (Song et al. 2019), FiGNN (Li et al. 2019), AFN (Cheng, Shen, and Huang 2020), and SAM (Cheng and Xue 2021), which automatically learn high-order feature interactions.

Then, we study a set of representative two-stream CTR models as introduced in the related work section.

Implementation We reuse the baseline models and implement our models based on FuxiCTR (Zhu et al. 2021), an open-source CTR prediction library[2]. Our evaluation follows the same experimental settings with AFN (Cheng, Shen, and Huang 2020), by setting the embedding dimension to 10, batch size to 4096, and the default MLP size to [400, 400, 400]. For DualMLP and FinalMLP, we tune the two MLPs in 1~3 layers to enhance stream diversity. We set the learning rate to $1e - 3$ or $5e - 4$. We tune all the other hyper-parameters (e.g., embedding regularization and dropout rate) of all the studied models via extensive grid search (about 30 runs per model on average). We note that through optimized FuxiCTR implementation and sufficient hyper-parameter tuning, we obtain much better model performance than what was reported in (Cheng, Shen, and Huang 2020)[3]. Thus, we report our own experimental results instead of reusing theirs to make a fair comparison. To promote reproducible research, we open sourced the code and running logs of FinalMLP and all the baselines used.

MLP vs. Explicit Feature Interactions

While feature interaction networks have been widely studied, there is a lack of comparison between MLP and well-designed feature interaction networks. Previous work proposes many explicit feature interaction networks, e.g., cross network (Wang et al. 2017), CIN (Lian et al. 2018), AutoInt (Song et al. 2019), and AFN (Cheng, Shen, and Huang 2020), to overcome the limitation of MLP in learning high-order feature interactions. Yet, most of these studies fail to directly compare explicit feature interaction networks with MLP (a.k.a, DNN or YouTubeDNN (Covington, Adams, and Sargin 2016)) alone, but only evaluate the effectiveness of two-stream model variants (e.g., DCN, xDeepFM, and AutoInt+) against MLP. In this work, we make such a comparison in Table 3. We enumerate the representative methods that are used for first-order, second-order, third-order, and higher-order feature interactions. Surprisingly, we observe that MLP can perform neck to neck with or even outperform the well-designed explicit feature interaction networks. For example, MLP achieves the best performance on Criteo, MovieLens, and Frappe, while attaining the second-best performance on Avazu, with only an AUC gap of 0.02 points compared to SAM. The observation is also consistent with the results reported in (Wang et al. 2021), where a well-tuned MLP model (i.e., DNN) is shown to obtain comparable performance with many existing models.

---

[2]https://fuxictr.github.io
[3]https://github.com/WeiyuCheng/AFN-AAAI-20/issues/11

Dataset	Metric	WideDeep	DeepFM	DCN	xDeepFM	AutoInt+	AFN+	DeepIM	MaskNet	DCNv2	EDCN	DualMLP	FinalMLP
Criteo	AUC	81.38	81.38	81.39	81.39	81.39	81.43	81.40	81.39	81.42	<u>81.47</u>	81.42	**81.49**
	Std	5.7e-5	8.0e-5	4.9e-5	9.5e-5	1.4e-4	5.9e-5	5.9e-5	1.3e-4	2.0e-4	6.6e-5	5.6e-4	1.7e-4
Avazu	AUC	76.46	76.48	76.47	76.49	76.45	76.48	76.52	76.49	76.54	76.52	<u>76.57</u>	**76.66**
	Std	5.4e-4	4.4e-4	1.2e-3	4.1e-4	5.2e-4	3.7e-4	9.2e-5	2.6e-3	4.7e-4	3.0e-4	3.5e-4	4.9e-4
MovieLens	AUC	96.80	96.85	96.87	96.97	96.92	96.42	96.93	96.87	96.91	96.71	<u>96.98</u>	**97.20**
	Std	3.2e-4	1.6e-4	5.5e-4	9.0e-4	4.4e-4	5.8e-4	5.8e-4	2.8e-4	3.6e-4	3.4e-4	4.3e-4	1.8e-4
Frappe	AUC	98.41	98.42	98.39	98.45	98.48	98.26	98.44	98.43	98.45	<u>98.50</u>	98.47	**98.61**
	Std	7.9e-4	1.6e-4	3.1e-4	3.7e-4	7.9e-4	1.4e-3	6.3e-4	5.7e-4	4.3e-4	5.1e-4	3.5e-4	1.7e-4

Table 2: Performance comparison of two-stream models for CTR prediction. The best results are in bold and the second-best results are underlined.

Class	Model	Criteo	Avazu	MovieLens	Frappe
First-Order	LR	78.86	75.16	93.42	93.56
Second-Order	FM	80.22	76.13	94.34	96.71
	AFM	80.44	75.74	94.72	96.97
	FFM	80.60	76.25	95.22	97.88
	FwFM	80.63	76.02	95.58	97.76
	FmFM	80.56	75.95	94.65	97.49
Third-Order	HOFM(3rd)	80.55	76.01	94.55	97.42
	CrossNet(2L)	79.47	75.45	93.85	94.19
	CrossNetV2(2L)	81.10	76.05	95.83	97.16
	CIN(2L)	80.96	76.26	96.02	97.76
Higher-Order	CrossNet	80.41	75.97	94.40	95.94
	CrossNetV2	81.27	76.25	96.06	97.29
	CIN	81.17	76.24	<u>96.74</u>	97.82
	AutoInt	81.26	76.24	96.63	<u>98.31</u>
	FiGNN	<u>81.34</u>	76.22	95.25	97.61
	AFN	81.07	75.47	96.11	98.11
	SAM	81.31	**76.32**	96.31	98.01
	MLP	**81.37**	<u>76.30</u>	**96.78**	**98.33**

Table 3: Performance comparisons between MLP and explicit feature interaction networks. The best results w.r.t. AUC are in **bold** and the second-best results are <u>underlined</u>.

Overall, the strong performance achieved by MLP indicates that, despite its simple structure and weakness in learning multiplicative features, MLP is very expressive in learning feature interactions in an implicit way. This also partially explains why existing studies tend to combine both explicit feature interaction networks with MLP as a two-stream model for CTR prediction. Unfortunately, its strength has never been explicitly revealed in any existing work. Inspired by the above observations, we make one step further to study the potential of an unexplored model structure that simply takes two MLPs as a two-stream MLP model.

DualMLP and FinalMLP vs. Two-Stream Baselines

Following existing studies, we make a thorough comparison of representative two-stream models as shown in Table 2. From the results, we have the following observations:

First, we can see that two-stream models generally outperform the single-stream baselines reported in Table 3, especially the single MLP model. This conforms to existing work, which reveals that two-stream models can learn complementary features and thus enable better modeling for CTR prediction.

Second, the simple two-stream model, DualMLP, performs surprisingly well. With careful tuning of MLP layers of the two streams, DualMLP can achieve comparable or even better performance compared to the other sophisticated two-stream baselines. To the best of our knowledge, the strong performance of DualMLP has never been reported in the literature. In our experiments, we found that increasing stream network diversity by setting different MLP sizes in two streams can improve the performance of DualMLP. This motivates us to further develop an enhanced two-stream MLP model, FinalMLP.

Third, through our pluggable extensions on feature selection and fusion, FinalMLP consistently outperforms DualMLP as well as all the other compared two-stream baselines across the four open datasets. In particular, FinalMLP significantly surpasses the existing strongest two-stream models by 0.12 points (DCNv2), 0.23 points (xDeepFM), and 0.11 points (AutoInt+) in AUC on Avazu, MovieLens, and Frappe, respectively. This demonstrates the effectiveness of our FinalMLP. As of the time of writing, FinalMLP ranks the 1st on the CTR prediction leaderboards of PapersWithCode[4] and BARS[5] (Zhu et al. 2022) on Criteo.

Ablation Studies

In this section, ablation studies are shown to investigate the effects of the important designs of FinalMLP.

Effects of Feature Selection and Bilinear Fusion Specifically, we compare FinalMLP with the following variants:

- **DualMLP**: the simple two-stream MLP model that simply takes two MLPs as two streams.
- **w/o FS**: FinalMLP without the stream-specific feature selection module.
- **Sum**: Using the summation fusion in FinalMLP.
- **Concat**: Using the concatenation fusion in FinalMLP.
- **EWP**: Using the Element-Wise Product (i.e., Hadamard product) fusion in FinalMLP.

---

[4]https://paperswithcode.com/sota/click-through-rate-prediction-on-criteo

[5]https://openbenchmark.github.io/BarsCTR/leaderboard/criteo_x1.html

(a) Avazu   (b) MovieLens   (c) Frappe
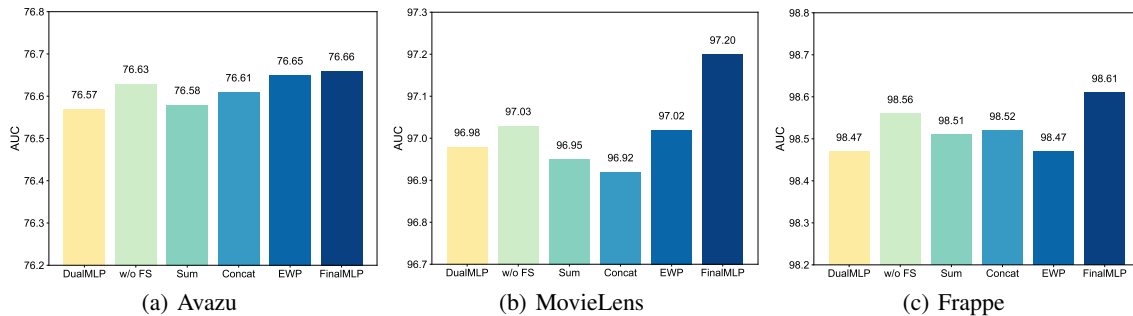
Figure 2: The ablation study results of FinalMLP.

#Heads (k)	Criteo	Avazu	MoiveLens	Frappe
1	OOM	0.7649	0.9691	**0.9862**
5	0.8141	0.7661	0.9707	0.9851
10	0.8144	**0.7669**	**0.9724**	0.9849
50	**0.8148**	0.7657	0.9703	0.9841

Table 4: Bilinear fusion with different numbers of heads.

The ablation study results are presented in Figure 2. We can see that the performance drops when removing the feature selection module or replacing the bilinear fusion with other commonly used fusion operations. This verifies the effectiveness of our feature selection and bilinear fusion modules. In addition, we observe that the bilinear fusion plays a more important role than the feature selection since replacing the former causes more performance degradation.

Effect of Multi-Head Bilinear Fusion We investigate the effect of our subspace grouping technique for bilinear fusion. Table 4 shows the performances of FinalMLP by varying the number of subspaces (i.e, number of heads k) for bilinear fusion. OOM means that Out-Of-Memory error occurs in the setting. We found that using more parameters (i.e., smaller k) for fusion does not always lead to better performance. This is because an appropriate k can help the model learn stream-level feature interactions from multiple views while reducing redundant interactions, similar to multi-head attention. One can achieve a good balance between effectiveness and efficiency by adjusting k in practice.

Industrial Evaluation

We further evaluate FinalMLP in our production system for news recommendation, which serves millions of daily users. We first perform an offline evaluation using the training data from 3-day user click logs (with 1.2 billion samples). The AUC results are shown in Table 5. Compared to the deep BaseModel deployed online, FinalMLP obtains over one point improvement in AUC. We also make a comparison with EDCN (Chen et al. 2021), a recent work that enhances DCN (Wang et al. 2017) with interactions between two-stream networks. FinalMLP obtains additional 0.44 points improvement in AUC over EDCN. In addition, we test the end-to-end inference latency between receiving a user request and returning the prediction result. We can see that by applying our multi-head bilinear fusion, the latency can be

	BaseModel	EDCN	FinalMLP	
			#Heads=1	#Heads=8
AUC	71.78	72.22	72.83	72.93
ΔAUC	–	+0.44	+1.05	+1.15
Latency	45ms	–	70ms	47ms

Table 5: Offline results in production settings.

	Day1	Day2	Day3	Day4	Day5	Average
ΔCTR	1.6%	0.6%	1.7%	1.5%	2.4%	1.6%

Table 6: Online results of a five-day online A/B test.

reduced from 70ms (using 1 head) to 47ms (using 8 heads), achieving the same level of latency with the BaseModel (45ms) deployed online. Moreover, the AUC result also improves slightly by selecting an appropriate number of heads. We finally report the results of an online A/B test performed on July 18th~22nd, where the results are shown in Table 6. FinalMLP achieves 1.6% improvement in CTR on average, which measures the ratio of users' clicks over the total impressions of news. Such an improvement is significant in our production systems.

Conclusion and Outlook

In this paper, we makes the first effort to study a simple yet effective two-stream model, FinalMLP, that employs MLP in each stream for CTR prediction. To enhance the input differentiation of two streams and enable stream-level interaction, we propose stream-specific feature selection and multi-head bilinear fusion modules that are pluggable to improve the model performance. Our evaluation on four open datasets and in industrial settings demonstrates the strong effectiveness of FinalMLP. We emphasize that the surprising results of FinalMLP question the effectiveness and necessity of existing research in explicit feature interaction modeling, which should attract the attention of the community. We also envision that the simple yet effective FinalMLP model could serve as a new strong baseline for future developments of two-stream CTR models. Moreover, it is also an interesting future work to plug our feature selection and bilinear fusion modules into more two-stream CTR models.

Acknowledgments

References

Blondel, M.; Fujino, A.; Ueda, N.; and Ishihata, M. 2016. Higher-Order Factorization Machines. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 3351–3359.

Chen, B.; Wang, Y.; Liu, Z.; Tang, R.; Guo, W.; Zheng, H.; Yao, W.; Zhang, M.; and He, X. 2021. Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 3757–3766.

Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys)*, 7–10.

Cheng, W.; Shen, Y.; and Huang, L. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 3609–3616.

Cheng, Y.; and Xue, Y. 2021. Looking at CTR Prediction Again: Is Attention All You Need? In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1279–1287.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 191–198.

Guan, L.; Xiao, X.; Chen, M.; and Cheng, Y. 2021. Enhanced Exploration in Neural Feature Selection for Deep Click-Through Rate Prediction Models via Ensemble of Gating Layers. *arXiv preprint*, abs/2112.03487.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1725–1731.

He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; and Candela, J. Q. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (ADKDD)*, 5:1–5:9.

Huang, T.; Zhang, Z.; and Zhang, J. 2019. FiBiNET: Combining Feature Importance and Bilinear Feature Interaction for Click-Through Rate Prediction. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 169–177.

Juan, Y.; Zhuang, Y.; Chin, W.; and Lin, C. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 43–50.

Li, Y.; Wang, N.; Liu, J.; and Hou, X. 2017. Factorized Bilinear Models for Image Recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2098–2106.

Li, Z.; Cui, Z.; Wu, S.; Zhang, X.; and Wang, L. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 539–548.

Lian, J.; Zhou, X.; Zhang, F.; et al. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*, 1754–1763.

Lin, T.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear CNN Models for Fine-Grained Visual Recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 1449–1457.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 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 (KDD)*, 1930–1939.

Pan, J.; Xu, J.; Ruiz, A. L.; Zhao, W.; Pan, S.; Sun, Y.; and Lu, Q. 2018. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, 1349–1357.

Pechuán, L. M.; Ponce, H.; and de Lourdes Martínez-Villaseñor, M. 2016. Feature Selection Methods Evaluation for CTR Estimation. In *Fifteenth Mexican International Conference on Artificial Intelligence (MICAI)*, 57–62.

Rendle, S. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, 995–1000.

Rendle, S.; Krichene, W.; Zhang, L.; and Anderson, J. R. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Fourteenth ACM Conference on Recommender Systems (RecSys)*, 240–248.

Richardson, M.; Dominowska, E.; and Ragno, R. 2007. Predicting Clicks: Estimating the Click-Through Rate for New Ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, 521–530.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, 1161–1170.

Sun, Y.; Pan, J.; Zhang, A.; and Flores, A. 2021. FM2: Field-matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference (WWW)*, 2828–2837.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 5998–6008.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 11th International Workshop on Data Mining for Online Advertising (ADKDD)*, 12:1–12:7.

Wang, R.; Shivanna, R.; Cheng, D.; Jain, S.; Lin, D.; Hong, L.; and Chi, E. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021 (WWW)*, 1785–1797.

Wang, Z.; She, Q.; and Zhang, J. 2021. MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask. *arXiv preprint arXiv:2102.07619*.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *The Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI)*, 3119–3125.

Yu, F.; Liu, Z.; Liu, Q.; Zhang, H.; Wu, S.; and Wang, L.

2020. Deep Interaction Machine: A Simple but Effective Model for High-order Feature Interactions. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM)*, 2285–2288.

Zhang, W.; Qin, J.; Guo, W.; Tang, R.; and He, X. 2021. Deep Learning for Click-Through Rate Estimation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 4695–4703.

Zhu, J.; Dai, Q.; Su, L.; Ma, R.; Liu, J.; Cai, G.; Xiao, X.; and Zhang, R. 2022. BARS: Towards Open Benchmarking for Recommender Systems. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2912–2923.

Zhu, J.; Liu, J.; Yang, S.; Zhang, Q.; and He, X. 2021. Open Benchmarking for Click-Through Rate Prediction. In *The 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 2759–2769.