ResFusion: A Residual Learning Based Fusion Framework for CTR Prediction

Junmei Bao[1], Yangguang Ji[1], Yonghui Yang[1], Le Wu[1,2(✉)], and Ruiji Fu[2]

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
hfut.baojunmei@gmail.com, jyguang1997@gmail.com, yyh.hfut@gmail.com, lewu.ustc@gmail.com

[2] State Key Laboratory of Cognitive Intelligence, iFLYTEK, Hefei, People's Republic of China
rjfu@iflytek.com

Abstract. CTR prediction tasks deal with the problem of evaluating the probability of users clicking on products, and have been widely deployed in many online recommendation and advertising platforms. Mainstream CTR models can be divided into two categories: the traditional machine learning models (e.g., GBDT [7]) that learn the linear feature combinations for prediction, and deep learning based algorithms (such as DeepFM [9]) for modeling the complex and sparse feature correlations. Some recent works proposed to fuse these two kinds of models for prediction. These fusion models either feed the intermediate results learned by one model into the second category or rely on the ensemble techniques to fuse two independently trained model outputs. In this paper, we propose a residual learning based fusion framework for CTR prediction. The key idea is that, we first train a model (e.g., GBDT), and let the second model (e.g., DeepFM) learn the residual part that can not be accurately predicted by the first model. The soundness of this framework is that: as the prediction power of these two kinds of models is complementary, it is easier to let the second model learn the residual output that can not be well captured by the first model. We show that our proposed framework is flexible and it is easier to train with faster convergence. Extensive experimental results on three real-world datasets show the effectiveness of our proposed framework.

Keywords: CTR prediction · Gradient Boosting Decision Tree (GBDT) · Deep Neural Network (DNN) · Models fusion

1 Introduction

With the prevalence of intelligent mobile devices, a huge volume of online transactions and browsing data has become available. Given huge number of items, Click Rate Prediction (CTR) has become a dominant element on these platforms. Specifically, CTR prediction focuses on predicting the likelihood of a user

Z. Dou et al. (Eds.): CCIR 2020, LNCS 12285, pp. 29–41, 2020.
https://doi.org/10.1007/978-3-030-56725-5_3

clicking an item, and the predicted items with larger probability can be displayed for users.

As CTR prediction is usually deemed as a classification problem in CTR prediction [4,8,9,11,14,16], current solutions could be divided into two categories: traditional machine learning based models [3,6,21] and deep learning based models [5,9,13,17,22]. As one of the proved most effective models among traditional machine learning based approaches, GBDT [7] constructs a tree structure by iteratively selecting the feature with the most significant statistical information gain, which is more conducive to automatically combining some dense numerical features, but it is impossible to learn very well for high-dimensional sparse category features [16]. Recently, DeepFM [9], as a representative deep learning based models, models the complex and hidden correlations between features for prediction, nevertheless, it's learning performance for dense digital features is not good enough. In fact, these two kinds of models are complementary, and jointly considering them would facilitate learning both the linear and non-linear features, in order to further enhance performance of either model.

The current solutions for combining the different models can be mainly classified into two categories: [1,11,18,24,25]: either to feed the intermediate results learned by one model into the second category [11,18,24,25], or to rely on the ensemble techniques to fuse two independently trained model outputs [1,18]. These models show the advantages of modeling both the linear and non-linear features together, and lead to better performance in practice. However, we argue that: since different models capture different characteristics of the data, instead of fusing them simplely, could we design a better fusion model that more explicitly utilizes the different prediction power of these two kinds of models? Considering the above characteristics, exploring how to use the respective advantages of GBDT and NN, and merging the two types of models effectively to solve the one-sided problem in feature learning seem critical.

In this paper, we propose a fusion framework learning strategy, based on the idea of ResNet [10], to improve the CTR prediction in real world data. The key idea is that, we first train a model (e.g., GBDT), and let the second model (e.g., DeepFM) learn the residual part that can not be accurately predicted by the first model. In fact, our proposed framework is an extension of the residual learning in the deep architecture design into model fusion for CTR prediction. Then, we analyze the soundness of this framework: as the prediction power of these two kinds of models is complementary, it is easier to let the second model learn the residual output that can not be well captured by the first model. We also show that our proposed framework is flexible and it is easier to train with faster convergence. Finally, extensive experimental results on three real-world datasets clearly show the effectiveness of our proposed model for the CTR prediction tasks.

2 Problem Definition and Related Works

2.1 Problem Definition

In a CTR prediction system, usually we can obtain the users' historical click records $D = \{(x_i, y_i)\}$ about the products. Let $x_i \in \mathbb{R}^d$ denote each sample with d features including numerical features and categorical features, and $y_i = \{0, 1\}$ denote observed label representing whether user clicks item. The CTR prediction task can be formulated as a supervised classification problem as follows:

Definition 1 (Click-Through Rate Prediction). *Given the training datasets $D_{train} = \{(x_{train}, y_{train})\}$, our goal is to learn a mapping function $f(x)$ that satisfies $\hat{y}_{train} = f(x_{train})$ reaching as much closer to y_{train} as possible. Then, for test datasets $D_{test} = \{x_{test}\}$, we compute $\hat{y}_{test} = f(x_{test})$ for user x_{test} to denote whether users will click on items.*

2.2 Related Works

In this section, we will mainly introduce the related works of the current CTR prediction tasks from the following three aspects: Traditional Machine Learning Models, Deep Neural Network Models and current Fusion Models.

Traditional Machine Learning Models. Logistic Regression is a generalized CTR prediction model that linearly combines each feature, it has been widely used in large-scale classification tasks due to its simplicity and low time complexity [20]. Apart from the linear combination of individual features, Factorization Machine (FM) [19] enumerates extra second-order cross information of all features and sends them into the model on the basis of LR. Field-aware Factorization Machines (FFM) [13] introduces the concept of field and assumes that each feature has different feature embeddings respectively for the different cross fields. Some other prevalent CTR prediction models derive from ensemble approaches. These approaches lie in three aspects: 1). **Boosting** works for the under-fitting models with high bias and low variance. For instance, AdaBoost [6] algorithm is one earliest classical implementation of the boosting algorithms, which is essentially a strong classifier constructed by a linear combination of multiple weak classifiers. Subsequently, some more effective gradient boosting methods, like GBDT [7], have been proposed as the promotion of AdaBoost algorithm by optimizing the loss function based on a negative gradient descent method. 2). **Bagging** approaches work for the instance of data with high variance but low bias. And Bagging could alleviate the high variance problems by bootstrap sampling from data. 3). **Stacking** could instead work for both variance and bias problems. It introduces a meta learner for aggregating heterogeneous component strong classifiers, which distinguishes the most from the former two approaches.

Deep Neural Network Models. Recently, many deep learning based CTR models have been proposed [5,9,17]. These models, focusing on how to more effectively model non-linear feature interactions, have been successfully applied to many industrial scenarios. Among them, Wide & Deep [5] can jointly learn by both the wide linear models and deep neural networks, which captures the low-order and high-order cross features. Besides, DeepFM [9] exceeds FM in extracting high-order combined features learnt by additional DNN parts, which can automatically combine high-order features without manual intervention in an end-to-end manner. Last but not least, xDeepFM [17] introduces a significant structure of the compressed interactive network (CIN) to generate feature interactions in an explicit fashion. Graph Convolutional Networks (GCNs) [2,23] iteratively encode graph structure and node features for node representation, which could capture the hidden feature interactions for CTR prediction.

Fusion Models. Since GBDT and NN models are suitable respectively for numerical features and categorical features, a growing number of methods emerge about how to fuse these two kinds of models for higher accuracy in prediction. These fusion models can be divided into the following two categories:

1) **Feature Fusion.** This kind of fusion models utilize the first model's results as additional features to train the second model. There are some fusion works directly combining the GBDT with NN on the feature level. In other words, they use one model's learning output results as additional feature inputs to feed into a second model with the same original data. For instance, we can extract leaf nodes of a pre-trained GBDT as a series of features input and then put them into a new model. Many works [11,18,25] have proved the effectiveness of this method, such as GBDT+LR [11] model which uses leaf nodes information trained by GBDT as combined features for LR training and GBDT2DNN [18] is a cascading fusion model that first trains a GBDT model, and the predicting score of GBDT is fed as an input feature into the DNN model. So this kind of fusion methods can be understood as a cascade process of feature engineering plus model learning.
2) **Prediction Fusion.** Another kind of fusion models usually intuitively combine the two model predictions by learning ensemble weights. In this way, $\overline{DNN + GBDT}$ [18] proposes to take a weighted average of prediction scores learnt separately from DNN and GBDT models sharing common training data and outputs the final probability score after an activation function. Another model named MTRecS-DLT [1] directly fuses the output scores of two single models in the ratio of 1:1 without using the *sigmoid* function.

In summary, traditional machine learning based models can accomplish linear feature combinations, and NN models use embedding strategy to solve complex feature intersections. Nevertheless, when they are faced with large-scale, heterogeneous data, one single model is no more effective because of their respective weaknesses. In addition, the existing fusion methods also have some notable shortcomings. For feature fusion models, only a single model is used as a feature

extraction process, failing to directly combine the complementary advantages of the two types of models. Besides, in terms of prediction fusion models, an additional ensemble frame is needed to fuse the results of two single models. The quality of the final prediction results depends excessively on the fusion ability of the additional ensemble frame. Therefore, based on the above characteristics, we propose a residual learning based fusion framework to alleviate the limitation of existing fusion methods. Figure 1 shows the differences between the existing fusion models and the ResFusion framework which we proposed.

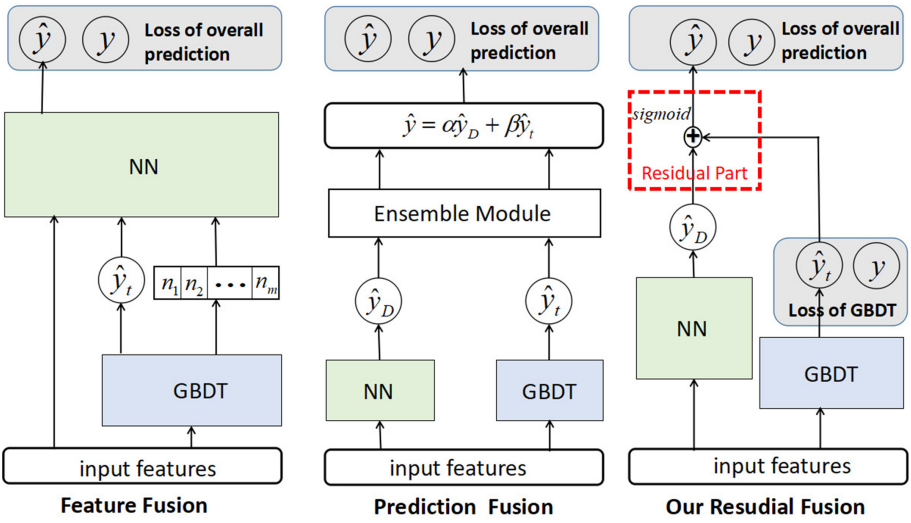


Fig. 1. The differences between our model and other fusion model

3 The Proposed Framework

In this section, we would introduce our proposed ResFusion framework for CTR prediction tasks in detail. We begin with the integral architecture, followed by the details of model components. At the end of this section, we would demonstrate the model training process and the discussion of our ResFusion framework.

3.1 Overall ResFusion Framework Architecture

The Fig. 2 shows the integral architecture of the ResFusion. By taking a feature set $X(x_1, x_2, x_3.....x_d)$ as input, it outputs the probability $\hat{y}$ that user would like to click the item (e.g., web pages or ads). The overall architecture of our model contains two main parts: the GBDT component and the DeepFM component. Specifically, by taking the related inputs, the GBDT outputs probability $\hat{y}_t$.

Then we can calculate the residual between the true label y and the predicted value of GBDT $\hat{y}_t$, and we called this value as res_t, which is the key of our model. Next, the residuals are sent into the DeepFM component as the new learning target, with the same input features as GBDT's. Then the DeepFM part would output residual prediction value $\hat{y}_D$, for complementary of GBDT component. The model will finally get the predicted value, which can be expressed as: $\hat{y} = \hat{y}_D + \hat{y}_t$. We detail each part used in our fusion model as follows:

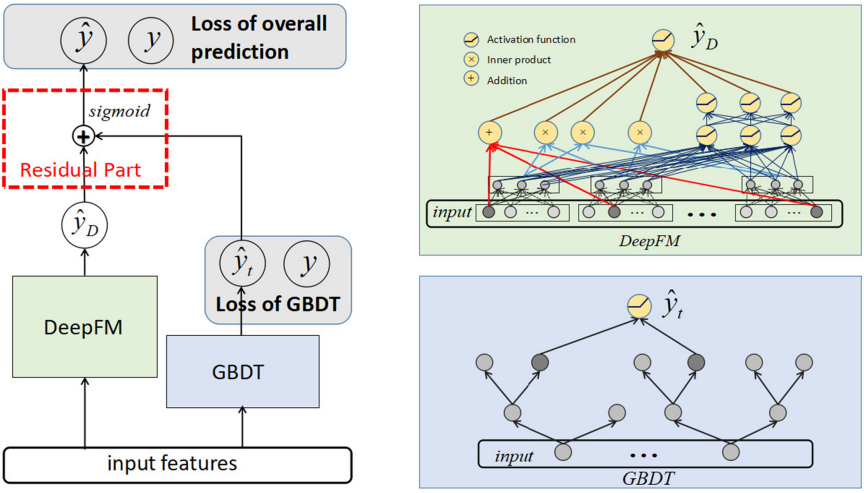


Fig. 2. The overall architecture of ResFusion

GBDT. GBDT is a decision tree algorithm based on the gradient boosting framework and can be seen on the right of the Fig. 2. "Gradient boost" means that each iteration process is to reduce the residual of the previous iteration, and a new weak classifier model is established in the direction of the gradient of the residual reduction. So the essence of GBDT algorithm can be expressed as the boosting method based decision tree:

$$F_M(x) = \sum_{m=1}^{M} T(x, \gamma_m) \tag{1}$$

Where $T(x, \gamma_m)$ represents the decision tree, γ_m represents the parameter of the tree, and M is the number of trees. And strong classifier $F_M(x)$ can be composed of multiple weak classifiers $T(x, \gamma_m)$ linear added. And by training a GBDT, we can get the prediction score $\hat{y}_t$.

Then we can compute the residual as:

$$res_t = y - \hat{y}_t \tag{2}$$

DeepFM. DeepFM is a neural network-based factorization machine (FM). Moreover, the model structure can be found in the right of the Fig. 2. This method contains two inner parts: FM part and DNN part. The FM part learns mainly from primary and second-order cross features as low-order features, while the deep part is a feed-forward neural network to extracts high-order cross features. FM and DNN share the standard features' input by linking with the common input layers and embedding layers. Finally, we combine the results of DNN y_{DNN} and FM y_{FM} and send them into an activate function. The final prediction result of DeepFM component is summed as:

$$\hat{y}_D(x_n) = y_{DNN}(x_n) + y_{FM}(x_n) \tag{3}$$

Then we sum the output of two components: $\hat{y}_t$, $\hat{y}_D$, and obtain the final prediction score $\hat{y} = \hat{y}_t + \hat{y}_D$.

3.2 Model Training

As our model contains two components, we first train a strong classifier called GBDT through the process of fitting the residuals by multiple iterations. The loss function for optimizing the tree model in every iteration process is:

$$\gamma_m = \arg\min_{\gamma} \sum_{n=1}^{N} \mathcal{L}(y_n, F_{m-1}(x_n) + T(x_n, \gamma)) \tag{4}$$

and the prediction score of GBDT $\hat{y}_t$ is used to calculate the residual with the true label y, and then the DeepFM try to fit this residual by optimizing the following formula:

$$W = \arg\min_{W} \sum_{n=1}^{N} \mathcal{L}(y_n, s(\hat{y}_D(x_n) + \hat{y}_t(x_n))) \tag{5}$$

where $s(x)$ is a *sigmoid* function, above two loss we all use *logloss* function to complete a binary classification task. γ and W represent the model parameters of GBDT and DeepFM. In practice, we use the *LightGBM* [15] to train a GBDT model, then we implement the DeepFM model with Pytorch[1] to train model parameters with mini-batch Adam.

3.3 Discussions

In this section, We would discuss our proposed framework from three aspects: convergence speed, model generalization ability and model flexibility.

Rapid Convergence. Our proposed model is trained on the basis of complementary of two single models. By fitting a new model on the remaining residuals between the true label and another model's output to learn what the former

[1] https://www.pytorch.org.

model failed to learn, the new model merely needs to learn less content until reaching convergence with relatively faster speed.

Model Generalization. ResFusion is designed under the problem setting with the input of the combination feature matrix F. When GBDT and DeepFM learn separately with the same input, they would focus on different content even in the same data according to their different learning methods. ResFusion's learning ability is no longer single and one-sided, and it can learn the hidden information more generally under the input data. Through the repeated joint learning of two completely different learning mechanisms, the optimal solution of the model can be obtained. So our fusion model finally proves to have better generalization and scalability.

Model Flexibility. ResFusion can also be understood as a result fusion model. Distinguished from other result fusion methods mentioned above, our model doesn't rely on the additional external fusion aggregation model for learning, rather, it's artfully sequentially links the two model as the fuse process during the model training process. So ResFusion can also be extended with feature fusion methods, which highlights the superior flexibility of our model.

In general, distinct from the plain feature fusion models, ResFusion uses the different learning capabilities of the two types of models more directly utilizing both the advantages from the models. Additionally, In our model, the latter (e.g., DeepFM) learns the remaining residual parts based on what the former (e.g., GBDT) has not learned, thus the speed of model convergence will be relatively accelerated. Specially, ResFusion has better generalization ability. Compared with the result fusion models, we fuse two methods naturally as an integral joint learning process and do not depend on the specific external aggregation method. Therefore ResFusion is also very flexible and can also be used in combination with other fusion methods mentioned formerly.

4 Experiments

In this section, we conduct extensive experiments on three real-world datasets to evaluate the effectiveness of our proposed fusion models.

Table 1. The statistics of the three datasets

Dataset	Total instances	Train	Test	Numerical features	Categorical features
Avazu	40 M	36 M	4 M	0	23
Cretio	45 M	40.5 M	4.5 M	13	26
ZhiHu	2 M	1.8 M	0.2 M	131	18

4.1 Experimental Settings

Datasets. To evaluate the effectiveness of our proposed fusion model, we conduct experiments on three public datasets: Avazu, Criteo and ZhiHu datasets:
1) Avazu. *Avazu*[2] comes from kaggle CTR prediction competition [12,14]. It consists of 40 M click logs arranged in chronological order along ten days.
2) Criteo. *Criteo*[3] as a famous and accessible benchmarking dataset widely used in CTR model evaluation [9,12]. It includes 45 M click records
3) ZhiHu. *ZhiHu*[4] derives from ZhiYuan 2019 artificial intelligence competition. The provided data consists of 2M instances of inviting users to answer questions. For the above three datasets, we first filled the null values in numerical features with 0 and categorical features with −1. After data pre-processing, we randomly split all records into train and test with the ratio of 9:1. The number of numerical features and categorical features for different datasets and other detailed dataset statistics are shown in Table 1.

Evaluation Metrics. We adopt two widely used evaluation metrics in experiments: AUC (Area Under ROC) and Logloss(cross entropy). AUC is used to evaluate the probability of ranking positive samples to be front while Logloss is used to measure the difference between predictions and true labels.

Baselines. We compare our proposed model with several state-of-the-art baselines for CTR prediction. We split all baselines into four groups: 1)Traditional machine learning models: LR [20], FM [19] and GBDT [7]; 2) Deep learning based models: DeepFM [9]; 3) Feature fusion models: GBDT+LR [11], GBDT2DNN [18], GBDT2DeepFM [18]; 4) Prediction fusion models: $\overline{GBDT + DeepFM}$ [1].

4.2 Overall Comparisons

In this section, we compare the overall performances of our proposed framework with other baselines. Specifically, Table 2 summarizes the AUC and Logloss values of various models on three datasets. We firstly analyze the single-models: LR only considers each feature's linear combination for CTR prediction, FM exceeds LR by combining the two features and obtaining the information of the second-order cross feature. GBDT can capture effective features and feature linear combinations efficient than LR by combining multiple weak classifiers. DeepFM performs better than FM, showing the effectiveness of the combination of DNN and FM. We find that DeepFM shows a better performance than GBDT on the Avazu dataset but not on the Cretio and ZhiHu datasets. The reason is, as we mentioned before, that shallow model is more suitable for dense numerical

[2] https://www.kaggle.com/c/avazu-ctr-prediction.
[3] https://www.kaggle.com/c/criteo-display-ad-challenge.
[4] https://biendata.com/competition/zhihu2019.

Table 2. AUC and Logloss comparisons for different models

Models	*Avazu*		*Cretio*		*ZhiHu*	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
LR	0.5453	0.4554	0.5690	0.5650	0.6122	0.5613
FM	0.7759	0.3820	0.7674	0.5052	0.7319	0.4102
GBDT	0.7608	0.3895	0.8009	0.4495	0.8390	0.3706
DeepFM	0.7852	0.3779	0.7959	0.4569	0.7712	0.3787
GBDT+LR	0.7634	0.3877	0.8025	0.4423	0.8405	0.3700
GBDT2DNN	0.7858	0.3761	0.8031	0.4417	0.8409	0.3699
GBDT2DeepFM	0.7863	0.3741	0.8037	0.4412	0.8417	0.3702
$\overline{GBDT + DeepFM}$	0.7860	0.3767	0.8022	0.4367	0.8411	0.3707
NNres+GBDT	0.7872	0.3726	0.8030	0.4379	0.8420	0.3702
GBDTRes+NN	**0.7921**	**0.3720**	**0.8065**	**0.4348**	**0.8676**	**0.3679**

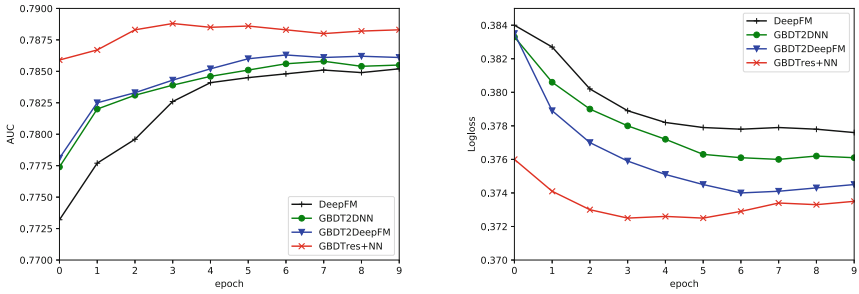
features and deep model is more suitable for sparse categorical features. Furthermore, we can find the difference on three datasets, the Avazu dataset has only categorical features. Then, we compare our model with other fusion models: for GBDT+LR, we take the output leaf nodes of GBDT as extra feature of data set to feed in LR; GBDT2DNN and GBDT2DeepFM are fed GBDT's predictions as extra features into DNN and DeepFM respectively. The three fusion models fuse GBDT with other models on feature-level and all exceed GBDT. Different from fusion on feature-level, $\overline{\text{GBDT} + \text{DeepFM}}$ model fuses GBDT and DeepFM on output-level by learning the weight parameters for two outputs for final predictions. Compared with other fusion models, our GBDTRes+NN model which is based on ResFusion framework consistently achieves best performance on both evaluation metrics. On *ZhiHu* dataset, our model improves best fusion baselines by 2.59% and 0.2% on AUC and Logloss, respectively. Based on the analysis of above experimental results, we could empirically conclude that our proposed ResFusion framework outperforms all baselines.

4.3 Detailed Model Analysis

In this subsection, we would like to give a detailed analysis of our proposed ResFusion framework and show the effectiveness of our fusion strategy.

Convergence Speed Analysis. We logged the convergence process of our GBDTRes+NN and other NN (DeepFM)-based models to verify that our model has faster convergence speed. Figure 3 shows the convergence process of AUC and Logloss values on the Avazu dataset. We find that our model achieves convergence at the second epoch. Compared with the deepFM model, which mainly requires nearly six epochs to converge, it is faster by nearly four epochs. Compared with the other two fusion models, our model is also faster by about two epochs. The reason is that our fusion model is based on residual learning, the

DeepFM module only needs to fit the residual part which GBDT did not learn very well, so it can achieve convergence rapidly.



(a) AUC of different models (b) Logloss of different models

Fig. 3. The convergence speed comparison on various fusion models

Table 3. AUC and Logloss comparisons with different number of iterations K.

Residual iteration	*Avazu*		*Cretio*		*ZhiHu*	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
K = 0	0.7608	0.3895	0.8011	0.4495	0.8390	0.3706
K = 1	0.7921	0.3720	0.8069	0.4350	0.8676	0.3679
K = 2	**0.7925**	**0.3717**	**0.8073**	**0.4341**	**0.8684**	**0.3679**
K = 3	0.7922	0.3720	0.8071	0.4351	0.8680	0.3680

Model Generalization Analysis. In this part, We verify the generalization of our proposed framework by setting different number of residual learning K. The experiment results can be observed in Table 3. In the verification experiment, we choose GBDT as the initial model so our fusion model can be seen as a single GBDT model when $K = 0$. Then the results of $K = 1$ mean that we use the DeepFM to fit the residual values of real-labels and the predictions of GBDT, called GBDTRes+NN. The improvement of AUC over the single model (GBDT) is 3.13%. After that, we feed the predictions of the first fusion model GBDTRes+NN's into the GBDT to fit the residual again when $K = 2$. We can do residual learning in an iterative way at different K. According to the experimental results, when $K = 2$, our strategy reaches the best, which means that through the first two residual learning, each model has already fully learned the residual part that another model does not learn. So as K increases to 3 from 2, the performance of the fusion model can no longer be improved, and may even result in over-fitting which leads to suboptimal results.

5 Conclusion

In order to alleviate the challenge that the existing CTR models cannot fully learn from data with both sparse category and dense numerical features, we propose a ResFusion framework which integrates GBDT and NN together by residual learning. It gains performance improvement for these advantages: 1) Compared with existing fusion models, it can directly utilize the complementary advantages of the component models; 2) In the process of fusion, it does not depend on the specific fusion method, so it is more generalized; 3) Residual-based fusion methods can boost model convergence. We finally conduct extensive experiments on three real-world datasets and prove the effectiveness and efficiency of our model over current state-of-the-art models on the two main evaluations of AUC and Logloss.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China (Grant No. 61972125, U19A2079), the Fundamental Research Funds for the Central Universities (Grant No. JZ2020HGPA0114), Zhejiang Lab (No. 2019KE0AB04) and the Foundation of Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R., China (Grant No. COGOS-20190002).

References

1. Abedalla, A., et al.: MTRecS-DLT: multi-modal transport recommender system using deep learning and tree models. In: 2019 SNAMS, pp. 274–278. IEEE (2019). https://doi.org/10.1109/SNAMS.2019.8931864
2. Chen, L., Wu, L., Hong, R., Zhang, K., Wang, M.: Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: AAAI2020, vol. 34, pp. 27–34 (2020)
3. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: KDD2016, KDD 2016, p. 785–794. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2939672.2939785
4. Cheng, H., Cantú-Paz, E.: Personalized click prediction in sponsored search. In: WSDM (2010). https://doi.org/10.1145/1718487.1718531
5. Cheng, H.T., et al.: Wide & deep learning for recommender systems. https://doi.org/10.1145/2988450.2988454
6. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the Second European Conferenceon Computational Learning Theory (1995). https://doi.org/10.1006/jcss.1997.1504
7. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat., 1189–1232 (2001). https://doi.org/10.1214/aos/1013203451
8. Graepel, T., Borchert, T., Herbrich, R.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine (2010). https://doi.org/10.5555/3104322.3104326
9. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. https://doi.org/10.24963/ijcai.2017/239
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

11. He, X., et al.: Practical lessons from predicting clicks on ads at facebook. In: Proceedings of the Eighth International Workshop on Data Mining for Online Advertising, pp. 1–9 (2014). https://doi.org/10.1145/2648584.2648589
12. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 169–177 (2019). https://doi.org/10.1145/3298689.3347043
13. Juan, Y., Lefortier, D., Chapelle, O.: Field-aware factorization machines in a real-world online advertising system. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 680–688 (2017). https://doi.org/10.1145/3041021.3054185
14. Juan, Y., Zhuang, Y., Chin, W.S., Lin, C.J.: Field-aware factorization machines for CTR prediction. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 43–50 (2016). https://doi.org/10.1145/2959100.2959134
15. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances In Neural Information Processing Systems, pp. 3146–3154 (2017). https://doi.org/10.5555/3294996.3295074
16. Ke, G., Xu, Z., Zhang, J., Bian, J., Liu, T.Y.: DeepGBM: a deep learning framework distilled by GBDT for online prediction tasks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 384–394 (2019). https://doi.org/10.1145/3292500.3330858
17. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xDeepFM: combining explicit and implicit feature interactions for recommender systems. https://doi.org/10.1145/3219819.3220023
18. Ling, X., Deng, W., Chen, G., Zhou, H., Cui, L., Feng, S.: Model ensemble for click prediction in bing search ads (2017). https://doi.org/10.1145/3041021.3054192
19. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000. IEEE (2010). https://doi.org/10.1109/ICDM.2010.127
20. Richardson, M., Dominowska, E., Ragno, R.: Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of the 16th International Conference on World Wide Web, pp. 521–530 (2007). https://doi.org/10.1145/1242572.1242643
21. Trofimov, I., Kornetova, A., Topinskiy, V.: Using boosted trees for click-through rate prediction for sponsored search. In: Data Mining for Online Advertising and Internet Economy, pp. 1–6 (2012)
22. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. https://doi.org/10.1145/3124749.3124754
23. Wu, L., Sun, P., Fu, Y., Hong, R., Wang, X., Wang, M.: A neural influence diffusion model for social recommendation. In: SIGIR2019, pp. 235–244 (2019)
24. Yang, A.: A recommendation system based on fusing boosting model and DNN model. https://doi.org/10.32604/cmc.2019.07704
25. YuChin Juan, W.S.C., Zhuang, Y.: 3 Idiots' Approach for Display Advertising Challenge (2014). https://github.com/ycjuan/kaggle-2014-criteo/