ResxDeepFM - Residual Net for Deep Recommender Systems

Doron Laadan , Maor Reuben , Bracha Shapira
Ben-Gurion University of the Negev
{laadan, maorreu}@post.bgu.ac.il, bracha.shapira@gmail.com,

Abstract

In the research area of recommender systems, Click-Through Rate (CTR) prediction, which aims to predict the probability of a user clicking on a given item, is an important task. Maximizing CTR prediction requires learning sophisticated feature interactions and capturing the latent user interest behind the user behavior data. Recently researchers have proposed several deep neural networks (DNN) based factorization models to learn these feature interactions both in low and high orders with great success. Some works even propose the automatic generations of new features using DNNs. However, due to long training time and vanishing gradients in deep neural networks, most works in the field use a shallow implementation of DNNs for their models with no more than five layers, which might limit their performances. In this work, we propose ResxDeepFM, the integration of a widely known neural network architecture, called Residual net, to the state-of-the-art xDeepFM model to improve its performances by allowing it to use deeper ANN. Additionally, we used our proposed approach as a *Deep Classifier* for the FGCNN network. We conduct comprehensive experiments on two real-world datasets to test our proposed methods and show their merits.

1 Introduction

Click-Through Rate (CTR) is a crucial task for recommender systems, which estimates the probability of a user to click on a given item[Chen *et al.*, 2016]. In an online advertising application, which is a billion-dollar scenario, the ranking strategy of candidate advertisements is by $CTR*bid$ where "bid" is the profit that the system receives once the advertisement is clicked on. In such applications, CTR prediction models' performance is one of the core factors determining the system's revenue.

It is essential for CTR prediction to learn implicit feature interactions behind user click behaviors [Cheng *et al.*, 2016]. Some feature interactions can be easily understood, thus can be designed by experts. However, most other feature interactions are hidden in data and challenging to identify so they can only be captured automatically by using machine learning tools, and as such, a wide range of models have been proposed to accomplish this task [Guo *et al.*, 2017; Juan *et al.*, 2016].

Generalized linear models have shown decent performance in practice despite being very simple. However, a linear model lacks the ability to learn feature interactions. The FM [Rendle, 2010] model, for example, learn pairwise feature interactions as an inner product of latent vectors between features and show very promising results. While in principle FM can model high-order feature interaction, usually only 2-degree feature interactions are considered due to high complexity.

In recent years due to their powerful ability to learn sophisticated feature interactions, deep neural networks (DNN) have been used more in the field of recommender systems. The Wide & Deep [Cheng *et al.*, 2016] and DeepFM [Guo *et al.*, 2017] networks are trained to capture both low and high-order feature interactions while DIEN [Zhou *et al.*, 2019] tries to model the user interest over time as well.

However, since useful interactions are always sparse, it is difficult for DNN to learn them effectively under a large number of parameters. Artificial features can improve the performance of deep models in real scenarios, but feature engineering is expensive and requires domain knowledge, making it impractical in different scenarios. Therefore, it is necessary to augment feature space automatically. The FGCCN [Liu *et al.*, 2019] and xDeepFM [Lian *et al.*, 2018] network try to approach this problem by automatically learning unique features by means resembling CNN's.

Most deep learning models systems suffer from the same disadvantage: The use of very shallow networks in their different components due to the vanishing gradient problem. In our work, we propose the integration of the residual network [He *et al.*, 2016] architecture into the state-of-the-art xDeepFM model in order to improve its performances by allowing for more deep network architectures to be used. We explore our proposed improvement in a similar way to xDeepFM. We combine both xDeepFM and our approach as *Deep Classifiers* for the FGCNN architecture to improve its results further. This, to the best of or knowledge, has not been tested yet.

The remainder of this paper is organized as follows: In Section 2, we provide a brief overview of related work; in Sec-

tion 3, we describe the proposed method that was used in this work; in Section 4, we describe the datasets and evaluation methods used in our experiments; in Section 5, we present the results that we obtained from our experiments; in Section 6, we provide discussion that arises from our results and finally, in Section 7, we give our conclusions from this study, and we offer future research directions.

2 Related Work

In this section, We will first provide a brief background on Click-Through-Rate prediction. Next, we will briefly review the origin of residual networks as well as provide an overview of the two main approaches for CTR predictions, namely using classical recommender systems and using deep learning for recommender systems while focusing on xDeepFM since our work is based on it.

2.1 Click Through Rate

Click-through rate (CTR) is a metric that measures the number of clicks advertisers receive on their ads per number of impressions. CTR prediction plays an essential role in recommender system since achieving a high CTR is essential to pay-per-click(PPC) service success, because it directly affects both quality and how much pay is needed every time someone clicks a search ad [Chen *et al.*, 2016]. In many recommendation systems, the goal is to maximize the number of clicks, so the items returned to a user should be ranked by estimated CTR. Simultaneously, in other scenarios such as online advertising, it is also essential to improve revenue, so the ranking strategy can be adjusted as $CTR * bid$ across all candidates, where "bid" is the benefit the system receives if a user clicks the item. Typically CTR prediction is formulated as a binary classification problem [Liu *et al.*, 2019], and as such, a wide range of Classification methods are used to solve this task.

2.2 Residual network

Much of the success of Deep Neural Networks have been accredited to the addition of multiple layers. The intuition behind their function is that these layers progressively learn more complex features, and with more layers, a better network can be created. However, it was shown in [He *et al.*, 2016] that this increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. Residual networks aim to solve this problem and allow for a more deep implementation of networks by introducing an "identity shortcut connection" that skips one or more layers, as shown in Fig 1. Using this shortcut, stacking layers shouldn't degrade the network performance, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. Over the years, residual networks gain popularity in the research community, and several new architectures based on ResNet were introduced, such as ResNeXt [Xie *et al.*, 2017] and Densely Connected CNN [Huang *et al.*, 2017] to name a few. In our work, we try to combine the main idea of Resnet, i.e., the 'skip connection' to allow Deep learning-based recommender systems to use deeper networks.

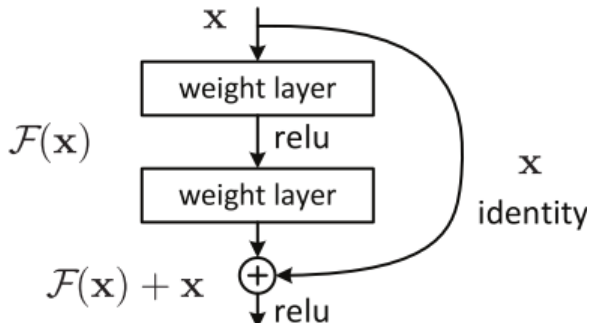


Figure 1: A residual block with skip connection

2.3 Classical Recommender Systems for CTR Prediction

For CTR predictions, the input features are usually sparse, categorical, continuous-mixed, and high-dimensional. For that reason, models like Logistic Regression (LR) with FTRL [McMahan *et al.*, 2013] are used due to their robustness, efficiency, and ability to manage and deploy quickly. However, these models lack the ability to learn feature interactions, so researchers are forced to design the pairwise feature interactions in the features space manually. Additionally, they can not generalize to unseen feature interactions in the training set.

Factorization Machine (FM) [Rendle, 2010] offers combination of regression and factorization models by introducing low-dimensional vectors for each feature and modeling feature interactions through inner product of feature vectors. By doing so, FM overcome LR problems via embedding each feature into a low dimension latent vector. FM have been shown to improve the ability of modelling feature interactions when data is sparse and does not require researchers to spend hours working on feature engineering.

In recent years several improvements to FM were introduced in the literature to improve the performances of recommendation systems. Field-Aware Factorization Machine (FFM) [Juan *et al.*, 2016] adds information to the data by transforming features to field related features. The transformation is done differently for numeric and categorical features. The proposed transformation enables each feature to have several latent vectors, and depending on the field of other features, one of them is used to do the inner product. It has been shown to improve the overall performance of the system, although at the cost of increased run-time and computing power due to the transformation needed to create the features to train FFM models.

Another example of improvement for FM is the Attentional Factorization Machine (AFM) [Xiao *et al.*, 2017]. Since not all feature interactions are equally useful and predictive, FM can be hindered by its modeling of all feature interactions

with the same weight. AFM uses attention mechanism (via a neural attention network) to improve pairwise interactions by learning the importance of each feature interaction from data itself. This leads to improved performances with a more straightforward structure and fewer model parameters than other methods.

Both AFM and FFM demonstrate that learning 2-degree interactions between features and performing feature selection by taking only the best interactions help the models to learn and improve performances. In our work, although we do not perform such feature selection, as deep learning frameworks rarely do, we do learn both implicit and explicit interaction between features to the k-degree and by using FGCCN we create meaningful feature automatically.

2.4 Deep Learning Recommender Systems for CTR Prediction

Deep learning techniques have achieved great success in many field of research such as computer vision, speech recognition and natural language processing and more. As a result, an increasing number of researchers are interested in employing DNNs for recommender systems[Lian *et al.*, 2018] since it can be used to avoid manually building up high-order cross features. Researchers can apply DNNs on field embedding, thus patterns from categorical feature interactions can be learned automatically from the data itself with little to no feature engineering. In a way, AFM can be seen as a deep learning recommendation systems since it adopts DNN to improve FM through the use of attentions network as describe before but we separate it from this section since it merely use the attention network to focus the FM itself.

The Wide & Deep model [Cheng *et al.*, 2016] jointly trains a wide model and a deep model where the wide model leverages the effectiveness of feature engineering using a simple logistic regression and the deep model learns implicit feature interactions through the use of embedding of the raw and transformed features. However, the wide component of the network still require features engineering which is expensive and requires expertise and the deep component can learn only pairwise feature interactions. The wide & Deep model is used as the base for other works such as DeepFM and xDeepFM and subsequently our work as well.

DeepFM [Guo *et al.*, 2017] provides a straightforward extension to the Wide & Deep model by replacing the logistic regression function of the wide component with an FM layer. The FM layer allows the model to avoid feature engineering and to learn implicit and explicit 2-degree interactions between features, unlike the Wide & Deep model. Also, research shows that by optimizing the entire graph together (both FM and DNN), DeepFM performs better than using the latent vectors from FM and then running them through the neural net as a secondary optimization. Despite its advantages, the DeepFM model suffers from some limitations. First, it can only learn 2-degree interactions between features and can only learn multiply interactions (for example it cannot learn $\sqrt{x_1} * x_2$). Second, no feature selection is performed before the FM component, which as been shown in FFM and AFM to increase FM performance. Finally, DeepFM mainly uses shallow DNN in the deep component and can miss complex interactions of features. According to the authors, an increasing number of hidden layers improves the models' performance at the beginning. However, their performance is degraded if the number of hidden layers keeps increasing because of overfitting. This problem arises in xDeepFM as well, and our main idea in this work is the use of residual networks to address this issue and allows such networks to utilize deeper architectures.

Another work that uses Deep learning for recommendation systems is Deep Interest Evolution Network (DIEN) [Zhou *et al.*, 2019], which uses an interest extractor layer to capture temporal interests from history behavior sequences using an auxiliary loss to supervise interest extracting at each step. The main idea beyond this work is two-fold; first, DIEN extracts latent temporal interests from explicit user behaviors. Second, DIEN models the interests evolving process of the user through the use of GRU with attention mechanism (AUGRU). Using AUGRU, DIEN can overcome the disturbance from interest drifting, and modeling for interest evolution helps the network to capture interest effectively, which further improves the performance of CTR prediction. Unlike DIEN, most deep learning frameworks cannot model the change in a user's interest over time, which might lead them to be susceptible to concept drift over time if they are not trained anew, as a person's preferences might change between sequential uses of the recommendation systems.

It was shown in the Wide & Deep model that artificial features are able to improve the performance of deep models, but creating these features is expensive and requires domain knowledge, which makes it impractical in different scenarios. The Feature Generation by Convolutional Neural Network (FGCNN) [Liu *et al.*, 2019] model was propose in order to augment feature space automatically and create new features with no domain knowledge and with minimal cost. FGCNN aims to reduce the learning difficulties of DNN models by identifying important features in advance. It is consists of two components: Feature Generation and Deep Classifier. An overview of FGCNN can be seen in Fig 2. In Feature Generation, a CNN+MLP structure is designed to identify and generate new important features from raw features. More specifically, CNN is performed to learn neighbor feature interactions, while MLP is applied to recombine them to extract global feature interactions. After Feature Generation, the feature space can be augmented by combining raw features and new features. In Deep Classifier, almost all state-of-the-art network structures can be applied to learn and predict based on the generated and raw features. In our work, we will explore the the use of xDeepFM as the Deep Classifier (which was not explored in [Liu *et al.*, 2019]) as well as our method as the Deep Classifier component of FGCNN.

eXtreme Deep Factorization Machine

Next, we will describe the eXtreme Deep Factorization Machine (xDeepFM) [Lian *et al.*, 2018] model in further detail since it is used as the base for our work. xDeepmFM architecture, which can be seen in Fig 3 follow a similar architecture structure to the Wide & Deep and DeepFM models, however, unlike previous models that learn interactions implicitly and at the bit-wise level, xDeepFM uses a novel Compressed In-

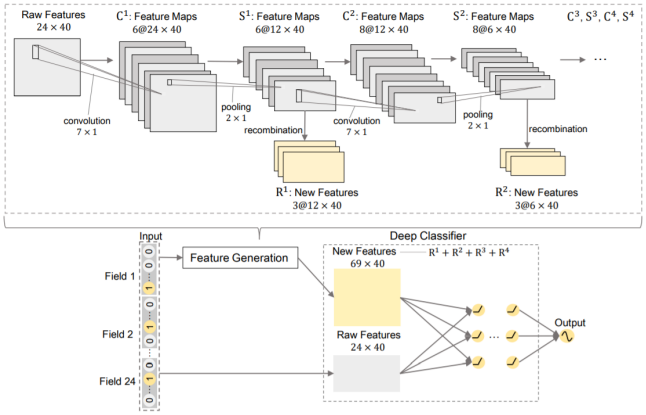


Figure 2: An overview of FGCNN as shown in [Liu *et al.*, 2019]. We will use xDeepFM and ResxDeepFM as Deep Classifiers.

teraction Network (CIN), which aims to generate feature interactions in an explicit fashion and at the vector-wise level. By combining a CIN and a classical DNN into one unified model, xDeepFM can learn certain bounded-degree feature interactions explicitly; on the other hand, it can learn arbitrary low- and high-order feature interactions implicitly.

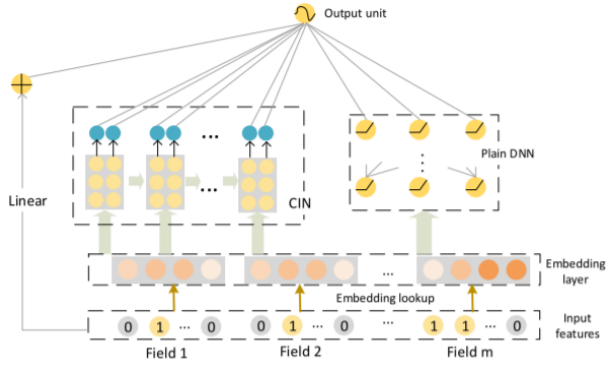


Figure 3: An overview of xDeepFM as shown in [Lian *et al.*, 2018].

The Compressed Interaction Network (CIN), is designed with the following considerations: interactions are applied at vector-wise level, not at the bit-wise level, high-order feature interactions are measured explicitly, and the complexity of the network will not grow exponentially with the degree of interactions. The structure of CIN, which can be seen in FIG 4 is very similar to the Recurrent Neural Network (RNN), where the outputs of the next hidden layer are dependent on the last hidden layer and additional input. Additionally, CIN has strong connections with the well-known Convolutional Neural Networks (CNNs) in computer vision through the use of intermediate tensor as a filter in its architecture. The filter moves across the embedding matrix and is used to acquire a *feature map*. The paper further proves the space and time complexity of this novel network. Since CIN and plain DNNs can complement each other, an intuitive way to make the model stronger is to combine these two structures. xDeepFM does that and the final output is provide by $\hat{y} = \sigma(W_{linear}^T + W_{dnn}^T * X_{dnn}^K + W_{cin}^T * P^+ + b)$.

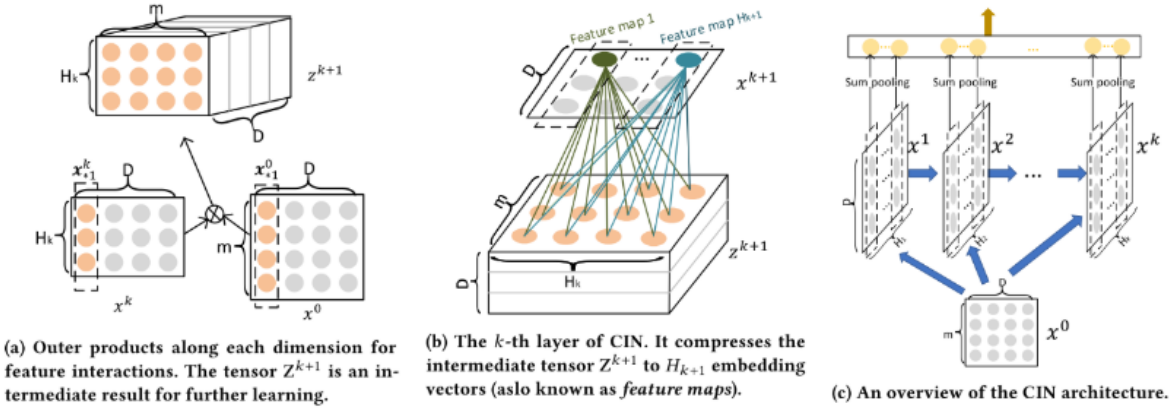
Assuming all fields are univalent, then when the CIN part's depth and feature maps are both set to 1, xDeepFM is a generalization of DeepFM by learning the linear regression weights for the FM layer. When the DNN part is removed, and at the same time a constant sum filter used for the feature map, then xDeepFM is downgraded to the traditional FM model.

xDeepFM main advantages are that it requires no feature engineering and can learn implicit and explicit K-degree interactions between features. However, it is not without disadvantages. First, The time complexity of the CIN network is high, second xDeepFM capture only multiply interactions as DeepFM and can add redundant interactions such as $x_1 * x_1 * x_1$ and finally like most deep architectures it uses only shallow networks for both CIN and DNN components due to time complexity and vanishing gradient problems. We will now propose our method for using the residual network to diffuse these issues.

3 The Proposed Method

We propose the ResxDeepFM, the integration of residual networks into the xDeepFM architecture. Our work tries to overcome a common disadvantage in deep recommend systems and xDeepFM specifically, the use of shallow network architectures. As mention before most deep recommender systems use a shallow network to overcome vanishing gradient problem, which limits their ability to learn more complex feature interactions. The problem has been shown to decrease with the use of residual networks. We first propose to use residual DNNs instead of the plain DNN component in xDeepFM architectures. We insert the 'skip connection' between layers, as seen in Fig 5, and allow the skip connection to be between a different number of layers, which is determined by an additional hyperparameter to the model, *skip freq*. For example, *skip freq* of one is the traditional Resnet architecture.

Next, we apply the same method to the CIN part of the xDeepFM, where between each matrix X (can be seen in Fig 4 (c)), there is a skip connection as well. The overview of our ResxDeepFM can be seen in Fig 6. It is similar to the xDeepFM architecture except each component, excluding the linear one, is replaces with its residual counterpart.



(a) Outer products along each dimension for feature interactions. The tensor Z^{k+1} is an intermediate result for further learning.

(b) The k-th layer of CIN. It compresses the intermediate tensor Z^{k+1} to H_{k+1} embedding vectors (aslo known as *feature maps*).

(c) An overview of the CIN architecture.

Figure 4: An overview of CIN as shown in [Lian *et al.*, 2018].

Figure 5: A skip connection

We hypothesize that using the residual network we can create a deeper DNN and CIN components without the risk of vanishing gradient problem. The deeper networks should, in theory, increase the model performance since better feature interactions can be learned.

Lastly, We propose using both xDeepFM and our new ResxDeepFM as a *Deep Classifiers* for the FGCNN network. The FGCNN will use the row features to automatically create new features that ResxDeepFM can later use to improve its performance. The idea to integrate Deep Classifiers is proposed in the original FGCNN paper, but it is never used with xDeepFM and naturally not with our new model.

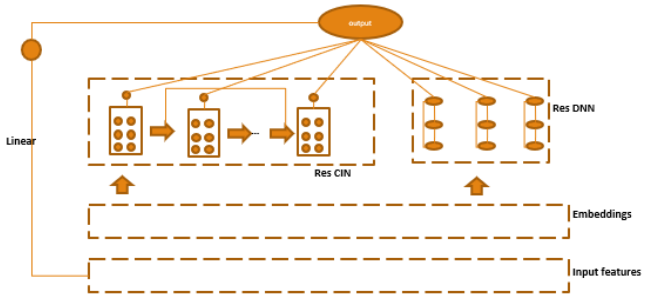


Figure 6: ResxDeepFM - similar to xDeepFM only all components are residual ones.

4 Evaluation

In this section, we describe the experiments we conducted. Our goals were to determine the following points:

- Does the inclusion of residual blocks help the DNN component to learn in a deeper setting (more layers)
- Does the inclusion of residual blocks help the CIN component to learn in a deeper setting (more layers)
- Can ResxDeepFM achieves better performance than the state-of-the-art-methods?
- Can xDeepFM improve the performance of FGCNN when used as a Deep Classifier?
- Can ResxDeepFM improve the performance of FGCNN when used as a Deep Classifier?

4.1 Experiment Setup

Method Implementation.

We implemented ResxDeepFM and its components using python 3.8 and TensorFlow 2.2. Our implementation follows the structure of layers and models in the DeepCTR python package and, in fact, can be used with DeepCTR API with minimal effort. Namely, we created two new layers, 'ResDNN' and 'ResCIN', and a new model 'ResxDeepFM,' which uses the above mention layers. Our implementation and code for experiments are publicly available [1].

DataSets.

We evaluate our proposed methods on the following two real-world datasets, both public and industrial:

Public dataset - Criteo. Criteo[2] is a famous benchmarking dataset for developing models predicting ad CTR, it was used in a Kaggle competition and is publicly accessible. Given a user and the page he is visiting, the goal is to predict the probability that he will click on a given ad. It includes 45 million users click records, and there are 13 continuous features and 26 categorical ones.

[1] https://github.com/DL1992/ResxDeepFM

[2] http://labs.criteo.com/downloads/download-terabyte-click-logs/

industrial dataset. Our industrial dataset is constructed by impression and click logs from an online display advertising company. The data was collected for one month from its systems and transformed to secure users' privacy (i.e., all fields are hashed). Given a user, the page he is visiting, and a given ad, the goal is to predict the probability that he will click on the ad. It includes around 30 million users click records, and there are a total of 23 features in it. Table 1 summarize the details of the datasets.

Dataset	Status	#Records	#Features	pos ratio
Criteo	Public	45M	39	50%
Industrial	Private	30M	23	50%

Table 1: Dataset statistics

Evaluation Methods and Metrics.
For both data sets, the labels in the test sets are not publicly available, so we split the available data into two sets for training and testing. For Criteo, we used an 80/20 split, and for the industrial dataset, the last 1M records were used as the test set. The models were trained on the training set and evaluated on the test set.

We use two common metrics for model evaluation: AUC (Area Under the ROC curve) [Hanley and McNeil, 1982] and Logloss (cross entropy) [Bishop, 2006]. These two metrics evaluate the performance from two different angels: AUC measures the probability that a positive instance will be ranked higher than a randomly chosen negative one. Logloss, in contrast, measures the distance between the predicted score and the true label for each instance. These two metrics are standard for evaluation in the field.

Baselines.
We compare our method to the following baselines: DeepFM, xDeepFM, and FGCNN. For DeepFM and FGCNN, we use the setting as describe in the the xDeepFm paper [Lian *et al.*, 2018] and FGCNN [Liu *et al.*, 2019]. In FGCNN, the Deep Classifier used this way is IPNN, as in the original paper.

For xDeepFM, we used the setting as described in [Lian *et al.*, 2018] 4.1.4 'Reproducibility' section for the Criteo dataset (in order to reproduce the results) as well as for the industrial dataset. Since we want to test our different components and compare them to xDeepFM components (i,e only DNN, and CIN), the same hyper-parameters from the original paper are used.

Experimental Plan.
The first experiment we conducted was to evaluate the Individual Components of our method. We want to know how the residual network affects each Component. The experiment is similar to section 4.2 in [Lian *et al.*, 2018]. We test the DNN, ResDNN, CIN, and ResCIN components separately using the method described above (both dataset, train-test split, parameter, and metrics). All baseline parameters are set to the same parameters as in [Lian *et al.*, 2018], but since the paper does not describe the number of epochs - we used the number set in the source code of xDeepFm and set the number of epochs for all experiment to 10. For the Res-variant of layers, we conducted several experiments with a different number of layers and skip connections.

The second experiment we conducted was to evaluate the Integrated Models. We compare ResxDeepFM to the baselines, and additionally, we evaluated two new combinations of models, FGCNN + xDeepFM and FGCNN + ResxDeepFM. The hyperparameters for all baseline are as mentioned and can be seen in Table 2. We used the best setting of each component from the first experiment in the integrated model experiment.

Model	Criteo	Industrial
General	opt=Adam batch=4096	
DeepFM	lr=0.001 L2R=0.0001 k=20 net=[400,400]	lr=0.001 L2R=0.0001 k=20 net=[400,400]
xDeepFM	lr=0.001 L2R=0.0001 k=10 dnn=[400,400] cin=[200,200,200]	lr=0.001 L2R=0.0001 k=10 dnn=[400,400] cin=[200,200,200]
ResxDeepFM	lr=0.001 L2R=0.0001 k=10 dnn=[400*8] cin=[200*5]	lr=0.001 L2R=0.0001 k=10 dnn=[400*5] cin=[200*4]
FGCNN	lr=0.001 conv=[9,9,9,9] kernel=[38,40,42,44] k=20 net=[4096,2048,1]	lr=0.001 conv=[7,7,7,7] kernel=[38,40,42,44] k=20 net=[4096,2048,1]

Table 2: hyperparameters setting - lr=learning rate,k=embedding size,opt=optimizer,L2R=l2 regularization of embedding,conv=shape of cnn kernels,net=shape of network

In order to perform the experiments on the amount of data we had, we did all pre-processing before the actual training and validation as well as creating specific data generator to load the data. Mostly, we imputed missing values with generic terms - numbers to minus one and categorical to null (The API we work with can handle these values through the use of embedding with little effect to the results). We hashed the Criteo dataset categorical features using label encoder and transform numeric values using MinMaxScalar. The code can been seen in the link provided before.

5 Results

5.1 Individual Components Results

In this section, we tested the individual components of xDeepFM (DNN and CIN) and compared them to our residual network variant ('ResDNN' and 'ResCIN'), this is part of our attempt to recreate the experiments in [Lian *et al.*, 2018]. Table 3 show the results of our experiments, the setting for DNN and CIN were chosen according to the setting in the xDeepFM paper, and the setting for ResDNN and ResCIN

are the settings which achieve the highest scores from our experiments. For the Crieto dataset, we observe a slight difference between the results reported in the paper and ours, and we believe this is the results of few things: first, for this section the paper neglect to describe all the hyper-parameters that were used (such as dropout rate, regularization) so we used the same parameters mention when describing the full xDeepFM model. Second, the number of epochs is not described, so we use ten epochs. Finally, the split records for the train and test sets might be different.

For both datasets, the best component is the CIN, as described in the original paper. We can see that our proposed approach does not improve the AUC of the individual components and might slightly harm the performance. This decrease in performance for our variants is less noticeable on the industrial dataset than on the Crieto dataset. However, we can see that using our approach, the best results are obtained by using a deeper network. Despite our hypothesis - our experiments showed the going much deeper in the residual variant with skip freq of one, meaning that between each layer exsit a skip connection, did not improve as much, and better results were obtained without adding many layers. We provide more analysis of the depth of the network in the next section.

Model Name	AUC	Logloss	Depth	# Neurons	Skip freq
Criteo					
DNN	0.789	0.452	2	400	-
ResDNN	0.779	0.511	8	400	1
CIN	**0.792**	0.449	3	200	-
ResCIN	0.781	0.483	5	200	1
Industrial					
DNN	0.809	0.64	2	400	-
ResDNN	0.804	0.651	5	400	1
CIN	**0.812**	0.53	3	200	-
ResCIN	0.806	0.57	4	200	1

Table 3: Performance of individual models on the datasets. Column Depth and skip freq indicates the best settings for each model

5.2 Integrated Models Results

Like xDeepFM, our approach integrated ResDNN and ResCIN into an end-to-end model. We tested whether combining them improve results like in the original xDeepFM work. Furthermore, we compare our approach to the baselines. Additionally, we not only tested ResxDeepFM but the integration of it and xDeepFM as *Deep Classifiers* to FGCNN as well, which, although proposed in [Liu *et al.*, 2019] to our knowledge has not been tested. The results are shown in Table 4.

The hyper-parameters for FGCNN are set according to [Liu *et al.*, 2019] and for DeepFM and xDeepFM, according to [Lian *et al.*, 2018]. ResxDeepFM hyper-parameters are set according to the best setting from the first experiment. Like in the first experiment, we can observe a slight difference from reported results to ours. This can occur for the same reasons mention before, such as a different number of epochs (which were not describe in the papers), different train/test split, for example. It is worth mentioning such differences also exist between the results reported in the FGCNN and xDeepFM papers.

The best model overall on both datasets is the FGCNN basic model. This complies with the results reported in the paper itself. Our proposed method shows that combining both components does improve the results like xDeepFM but did not improve upon the state-of-the-art and appears to have the lowest AUC among all baselines even when used as Deep classier to the FGCNN model. This leads us to believe that the skip connection of the residual architecture decreases the network's ability to learn implicit feature interactions properly via the CIN and DNN components. Combining xDeepFM with FGCNN also did not improve the performance of the individual networks.

Lastly, In figures 7 -8 we provide an analysis of using different layers depth in the network and using different skip freq for ResxDeepFM on both datasets. The number of neurons and other hyper-parameters are constant. For the Criteo dataset, we could afford to use a deeper network than in xDeepFM to achieve the best results. These results were, however, inferior to the baseline. In the industrial dataset, we can observe that deeper networks did not improve the performance, and in fact, only a slightly deeper network produces the best performances for our model than the baseline.

When testing the skip freq, we used different combinations of layers depths. Fig 8 shows the result when using the best depth setting, as reported in the paper. We can see that using more than skip freq of two severely hurt performances and that using a skip freq of one achieves the best results for our proposed approach.

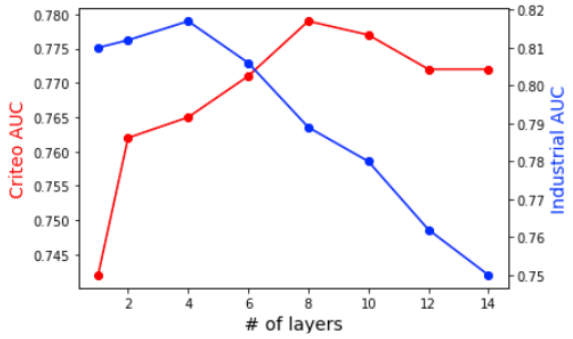


Figure 7: Impact of network layers on AUC performance.

6 Discussions

In this section, we provide further discussions about the results shown in the previous section. Although extensive experiments were done, we could not wholly reproduce the results given in[Lian *et al.*, 2018]. We consider several factors for this, the first being the different train/test split has only the ratio was given in the paper, it is likely we had different records in our test and train test, if the records are not balanced this could affect results - it may cause a cold start problem for example. Additionally, the number of epochs is significant to training networks, but since no information on

	Criteo		Industrial	
Model	AUC	Logloss	AUC	Logloss
DeepFM	0.786	0.472	0.823	0.51
FGCNN	**0.802**	0.447	**0.84**	0.49
xDeepFM	0.796	0.459	0.839	0.491
ResxDeepFM	0.779	0.521	0.817	0.561
FGCNN + xDeepFM	0.791	0.453	0.797	0.587
FGCNN + ResxDeepFM	0.783	0.542	0.789	0.601

Table 4: Overall performance of different models on datasets. The table shows the results of the best setting for networks

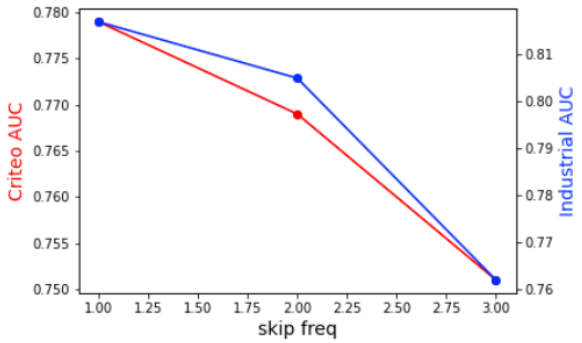


Figure 8: Impact of skip freq on AUC performance.

the number of epochs was given in the paper and since we lack the computing power, we perform ten epochs in all experiments, if the original paper performs more this might be a cause for difference. Lastly, we used the DeepCTR implementation of all networks - this implementation may be different from the original implementation, which might affect the results. In order to run the experinets we performs a slight data imputation to missing values - since no detail of this was mention in the papers we believe this effect the results as well.

The results show that combining xDeepFM or ResxDeepFM with FGCNN lead to decrease in model performances in all cases. This decrease is surprising since FGCNN report it can integrate with any model to improve its performances. One possible reason for this is that xDeepFM CIN component implicit features does not generalize well to the automatic features FGCNN creates, or that combining such large network architectures causes a vanishing/exploding gradient problem which decrease the learning. We do believe more experiments are to be performed before deciding that FGCNN can not be viewed as a general framework to enhance the existing neural networks by generating new features automatically.

As in all deep learning work, hyper-parameters tuning plays an essential part in reaching the best performance. Due to the time and computational power constraints we had, we could not perform many experiments on the different architecture to determine the best hyperparameters. Mostly, we used the parameters reported in the literature and change the hyper-parameters most important to ResxDeepFM, the depth, and skip freq. We believe that tuning the other hyper-parameters of the models such as dropout, regularization, and more could have resulted in better performances for our proposed method.

The main conclusion from our work is that integrating the residual network architecture into the xDeepFM model does not help improve results, and using a skip frequency of more than one significantly harms the performances of the model. A large skip frequency may harm the results since it causes the model to 'skip' meaningful feature interactions, limiting the model's ability to predict CTR. Additionally, using the residual architecture did not allow the models to use deep networks as we believe it would while improving results. We were surprised by this since, in the worst case, using skip connection should allow the network to learn the identity function between layers, and as such, using deeper networks should, if not improve, results not harm them. We believe in the case of CTR predictions, and unlike images where ResNet is used, it is harder for the network to learn this function given the features and task. Additionally, we think that using skip connection between CIN filters might decrease their benefit in learning implicit k-degree interactions, which harms the model once more.

7 Conclusion - Future work

In this work, we presented ResxDeepFM, the integration of a widely known Residual network, to the state-of-the-art xDeepFM model. ResxDeepFM allows the xDeepFM model, which uses shallow DNNs due to long training time and vanishing gradients, to use deeper ANNs and CIN layers, although with no improvement in results. Additionally, we explored the use of our proposed approach and xDeepFM as a *Deep Classifier* for the FGCNN network, which to the best of our knowledge, was not done before.

For future work, we plan to explore more combinations of Deep learning models (such as AFM and xDeepFM). We intend to explore further the use of residual networks and more advanced variances of ResNet to improve existing deep learning CTR prediction models.

References

[Bishop, 2006] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Chen *et al.*, 2016] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua. Deep ctr prediction in display advertising. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 811–820, 2016.

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

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[Hanley and McNeil, 1982] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[Juan *et al.*, 2016] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50, 2016.

[Lian *et al.*, 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1754–1763, 2018.

[Liu *et al.*, 2019] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. Feature generation by convolutional neural network for click-through rate prediction. In *The World Wide Web Conference*, pages 1119–1129, 2019.

[McMahan *et al.*, 2013] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[Xiao *et al.*, 2017] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[Zhou *et al.*, 2019] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5941–5948, 2019.