Research on Fusion of Deep Learning Models Based on Microservice Architecture

Hang Zhang (921254176@qq.com)
Changchun University of Science and Technology

Research Article

Keywords: Microservice Architecture, Deep Learning, CTR, DCN, DeepFM, xDeepFM

Posted Date: August 8th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1913303/v1

Research on Fusion of Deep Learning Models Based on Microservice Architecture

Hang Zhang

Changchun University of Science and Technology, Jilin Changchun 130022, China;

Corresponding author(s). E-mail(s): 921254176@qq.com; ORCID: 0000-0003-3348-9271

Abstract

Due to the differences in development languages and development frameworks, various difficulties are often faced in the process of integrating microservice architecture and deep learning models. This paper proposes a DND-P (Deep Model in Nacos Discovery Protocol) protocol to realize the rapid integration of deep learning models and microservice architectures. First, this paper selects three representative deep learning models DCN (Deep & Cross Network), DeepFM and xDeepFM in the field of CTR (Click Through Rate) as the research objects, and randomly samples 600,000 of them from the Criteo public data set. The model is trained with 10,000 sample sets to obtain a relatively stable model version. Second, use the Django framework to integrate the trained model as a web application. Finally, the DND-P protocol is directly referenced in Django to implement service registration in the microservice architecture, and the CTR is estimated at the business layer through OpenFeign. Through experiments, it is concluded that the integration of microservice architecture and deep learning model with DND-P protocol can break the barriers between the two, so as to quickly realize the integration of business applications.

Keywords: Microservice Architecture; Deep Learning; CTR; DCN; DeepFM; xDeepFM;

1 Introduction

With the widespread application of digital information technologies such as mobile Internet, big data, cloud computing, and artificial intelligence in all walks of life, the scale and data volume of software projects are increasing, and the complexity of software products is also increasing. Modern enterprise development requires not only a software development architecture that supports high concurrency and big data, but also rapid releases.Today, China is accelerating the construction of the digital economy, and major Internet companies are embarking on digital transformation. In the process of digital transformation, the selection of IT architecture has become more and more advanced, and most of them are based on cloud native concepts[1] combined with micro-service architecture for research and development digital platform.Among them, the representative technology is Spring Cloud Alibaba[2],This is mainly due to the complete technical components included in the Microservice Architecture System[3],including: service gateway, service registration, service discovery, load balancing, service degradation, service fuse, etc. Therefore, enterprises can quickly build personalized applications through the Microservice Architecture System.Secondly, Feng Zhiyong[4] and others conducted in-depth research on the role of Microservice Architecture in contemporary times, and concluded a series of problems faced by Microservice Architecture in application;It also expounds the opportunities and challenges brought by the emerging cloud computing platform and IOT platform to enterprises, and the challenges include how to integrate with the Microservice Architecture .

In addition, in the digital platforms of Internet e-commerce companies, search advertising and e-commerce advertising account for the highest proportion of profit models with online advertising as the mainstream.It is precisely because of its huge commercial value that it has become one of the main research objects in computational advertising, in which CTR (Click Through Rate, click-through rate) is an important indicator to measure the monetization ability of advertising.Today, the super computing power of cloud computing has enabled the rapid development of machine learning technology and deep learning technology, which are widely used in CTR prediction, and the prediction accuracy is also very high.However, how to integrate the CTR prediction model based on deep learning technology with the Microservice Architecture is currently a major problem faced by many Internet companies.Based on the above background, this paper proposes a DND-P (Deep Learning Model in Nacos Discovery Protocol) protocol, which aims to solve the difficulty of how to integrate deep learning models with Microservice Architecture.The main contents of this paper include the following points:

1) The principle of the deep learning model is explained. The Criteo data set provided by Kaggle is selected, and 600,000 data sets are extracted by random sampling technology to train the model, so as to obtain three relatively stable CTR prediction models (DCN, DeepFM, xDeepFM) .
2) DND-P (Deep Model in Nacos Discovery Protocol) protocol algorithm description.
3) Integration and testing of deep learning CTR prediction model and microservice architecture.

2 Microservice Architecture

2.1 Nacos Introduction

Nacos[5] is a dynamic service discovery, configuration management and service management platform open sourced by Alibaba Group that is easy to build cloud-native applications. In the Microservice Architecture Spring Cloud Alibaba, the governance of microservices is an indispensable technology. Among them, Nacos provides a set of simple and easy-to-use feature sets, which can help enterprises quickly realize dynamic service discovery, service configuration, service metadata and traffic management.The Microservice Architecture in this article is developed on the basis of Spring Cloud Alibaba, so Nacos is chosen as the registration and discovery management platform for microservices.

2.1 Architecture diagram

The microservice architecture diagram after integrating the deep learning model is shown in Figure 1:

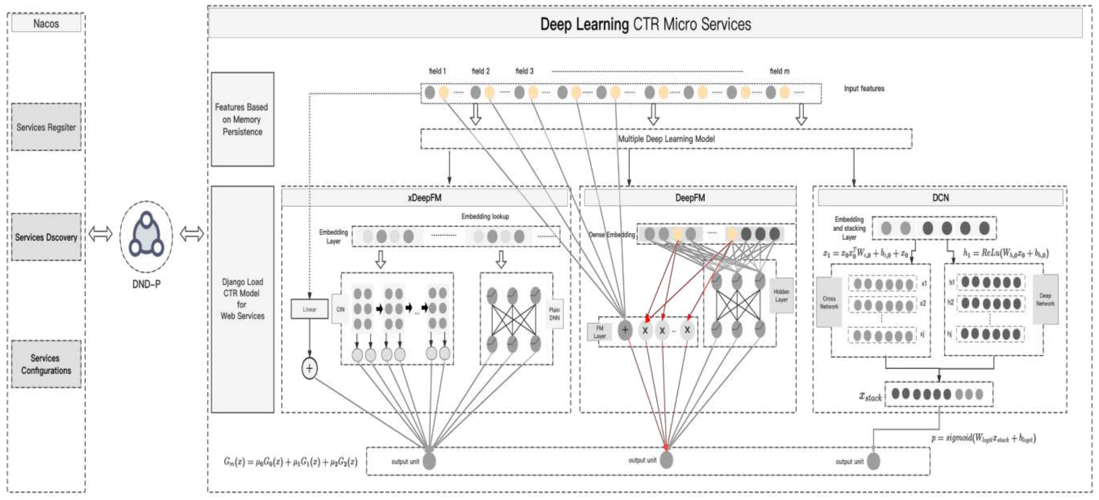
Nacos
Services Regsiter
Services Dscovery
Services Configurations
DND-P
Deep Learning CTR Micro Services
Features Based on Memory Persistence
Django Load CTR Model for Web Services
field 1
field 2
field 3
field m
Input features
Multiple Deep Learning Model
xDeepFM
Embedding lookup
Embedding Layer
Linear
CIN
Plain DNN
DeepFM
Dense Embedding
Hidden Layer
FM Layer
DCN
Embedding and stacking Layer
$x_1 = x_0 x_0^T W_{c,0} + b_{c,0} + x_0$
$h_1 = ReLu(W_{h,0} x_0 + b_{h,0})$
Cross Network
Deep Network
x1
x2
xj
h1
h2
hj
x_{stack}
$p = sigmoid(W_{logit} x_{stack} + b_{logit})$
$G_m(x) = \mu_0 G_0(x) + \mu_1 G_1(x) + \mu_2 G_2(x)$
output unit
output unit
output unit

Figure 1: Microservice Architecture Diagram After Integrating Deep Learning Model

- The left part in Figure 1 is the Nacos Server platform, whose main functions are service registration, service discovery, and service configuration management.
- The Deep Learning CTR Micro Services on the right is the three trained CTR deep learning prediction models, of which Input features, Mutiple Deep Learning Model are the input feature vectors of the three deep learning models DCN, DeepFM, xDeepFM and the weight parameters after training, And persist in the memory to ensure that the system can quickly output the results during the initial loading and calling process, and finally use the Django Web framework[6] to build a Web service for the CTR prediction model.
- The middle part is the DND-P protocol, which realizes the integration of the CTR deep learning prediction model and the microservice architecture.

The final prediction result in this paper is the fusion result of the above three CTR deep learning prediction models, as shown in Equation (1):

$$G_m(x) = \omega_0 G_0(x) + \omega_1 G_1(x) + \omega_2 G_3(x) \tag{1}$$

Defined as the function Gm(x) as a hypothetical function to perform the fusion prediction of the three models, Gm(x) can map continuous values between 0 and 1,where $G_i(x)$ represents the prediction result of the i-th CTR deep learning model, ω_i represents the weight coefficient of each model, and ω_i is defined as shown in Equation (2):

$$\omega_i = exp(G_i(x)) \div \sum_{k=1}^{n}(exp(G_k(x))) \tag{2}$$

The theoretical support is: since the result predicted by each deep learning model is linearly related to the random variable Y, the idea of multiple logistic regression is adopted, combined with the idea of the activation function softmax of the output layer in deep learning, and the two ideas are integrated. Thereby, the value of ω_i is calculated.

3 Deep Learning Model

3.1 DeepFM

DeepFM[7] enables the model to learn the interaction of low-order and high-order features. It integrates the FM model of embedding and the neural network of DNN, which capture low-order features and high-order features respectively, and finally combine them.The one-hot sparse matrix can make each feature map to a non-zero vector through the embedding of FM, thereby increasing the generalization ability of the model.However, FM cannot perform mining of high-order features.DNN can mine high-order features, thus making up for the shortcomings of FM.The structure of the DeepFM model is shown in Figure 2:

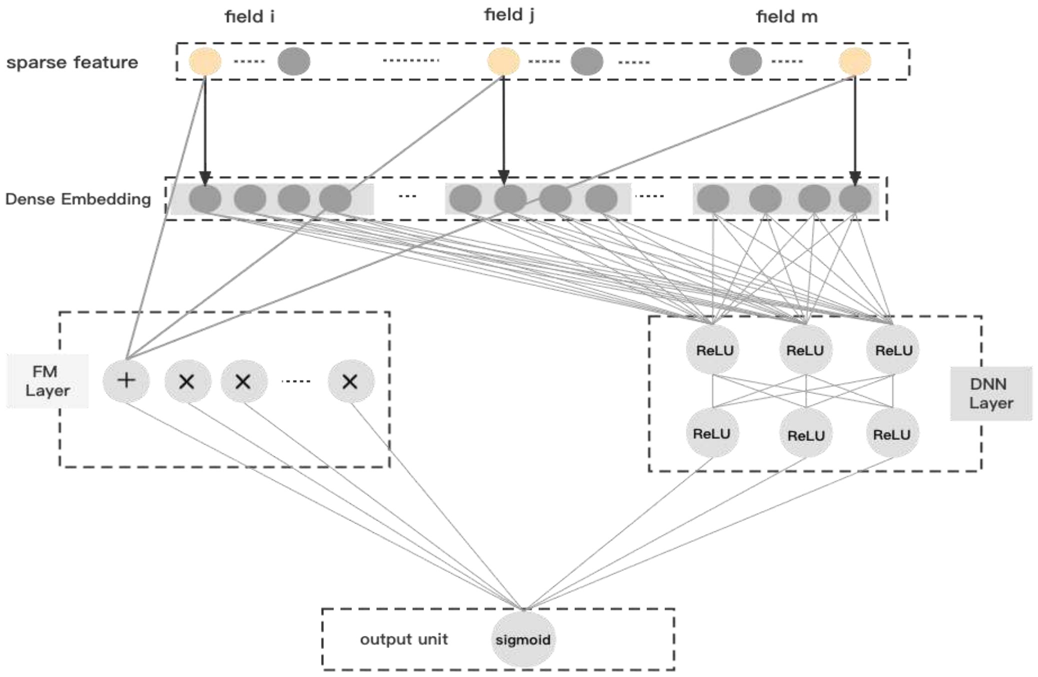

Figure 2: The Structure Of The DeepFM Model

Assuming that the outputs of the FM Layer and DNN Layer in the DeepFM model are yFM and yDNN, respectively, the final CTR prediction result of the DeepFM model is:

$$y = sigmoid(y_{FM} + y_{DNN}) \tag{3}$$

3.1.1 FM

The idea of the FM model is to consider the relationship between any two features after the LR (Logistic Regression) model. The initial definition of the model equation is as follows:

$$y_{FM} = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} x_i x_j \tag{4}$$

Where ω_{ij} is the cross weight of X_iX_j. The model introduces a second-order feature combination, but when X_iX_j=0, ω_{ij} =0; therefore, ω_{ij} eventually becomes 0 in the case of large-scale sparse features.To this end, the FM model can be improved as follows:

$$y_{FM} = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j \tag{5}$$

where <Vi,Vj> is the inner product of two vectors of length k, <Vi,Vj> as follows:

$$< v_i, v_j >= \sum_{f=1}^{k} v_{if} \times v_{jf} \tag{6}$$

The improved FM model is able to fully capture all first-order and second-order feature combinations, avoiding the inability of the model to train in the case of sparse data.Among (5),(6), ω_0 is the global bias value, and ω_i is the weight of X_i. $< v_i, v_j >$ is used to replace the original ω_{ij} to represent the weight of the second-order feature combination X_iX_j.This is the biggest difference between FM models and polynomial regression.As long as the combination of feature Xi and other arbitrary features has appeared, the corresponding embedding vector Vi can be learned through training. So when predicting, although the combination of Xi and Xj never appears in the training data, it can be used as a weight by calculating the inner product of their corresponding embedding vectors.

3.1.2 DNN

The DNN part is composed of multi-layer vi as a feedforward neural network to learn high-order feature combination information. However, since the original input features of CTR estimation are highly sparse, ultra-high-dimensional, mixed in types (continuous and discrete), and grouped by fields, an embedding layer is added before the ordinary feedforward neural network, and the feature groups of different input lengths are analyzed. compression. Another advantage of embedding is that the DNN part can share the embedding vector with the FM part, and there is no need to train the embedding vector separately. Suppose the output of the embedding layer is:

$$z^0 = (e_1, e_2, \cdots e_n) \tag{7}$$

Then take z as the input of the feedforward neural network, and its forward propagation process is as follows:

$$z^{m+1} = ReLU(\omega^m z^m + b^m) \tag{8}$$

Let the number of hidden layers of the feedforward neural network be h, and the final output of the DNN part is:

$$y_{DNN} = sigmoid(\omega^{h+1} z^h + b^{h+1}) \tag{9}$$

The FM component and the Deep component share the same embedding, which avoids complex and tedious feature engineering, and can effectively learn high- and low-order interactive features from the input features.

3.2 DCN(Deep&Cross Network)

The Deep Cross Network (DCN) model[8] is Google's follow-up research on Wide&deep[9], which replaces the Wide&deep part of Wide&deep with the crossover realized by a special network structure, automatically constructs limited high-order crossover features and learns the corresponding weights, eliminating the complicated manual cross-product step,as shown in Figure 3:

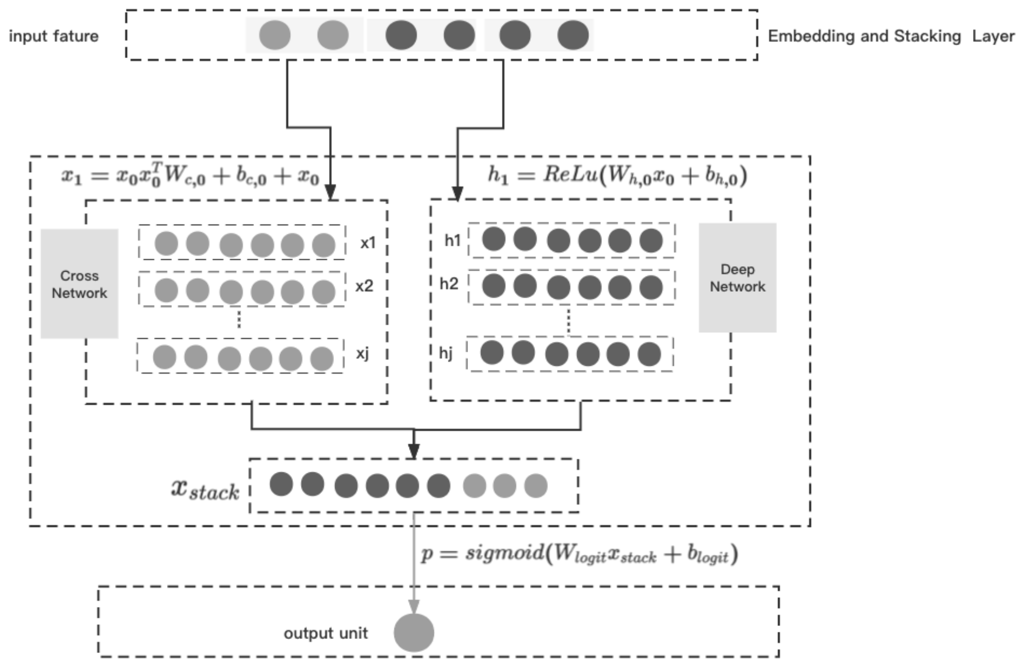
input fature
Embedding and Stacking Layer
$x_1 = x_0 x_0^T W_{c,0} + b_{c,0} + x_0$
$h_1 = ReLu(W_{h,0} x_0 + b_{h,0})$
Cross Network
x1
x2
xj
h1
h2
hj
Deep Network
x_{stack}
$p = sigmoid(W_{logit} x_{stack} + b_{logit})$
output unit

Figure 3: The Structure Of The DCN Model

The bottom layer of the DCN model is mainly divided into two layers: Embedding and Stacking. After embedding, the one-hot feature can be converted into a real dense vector of the specified dimension.The matrix parameters generated in the embedding process are trained together with other parameters in the network, and then the embedding vector and the normalized embedding vector are stacked as new input features.The equation is as follows:

$$x_0 = (x_{embed,1}^T, \cdots x_{embed,k}^T, x_{dense}^T) \tag{10}$$

In the above formula, x0 represents the input of the network; $x_{embed,k}^T$ represents the embedding vector; x_{dense}^T represents the normalized dense vector;In order to effectively apply the explicit intersection features, the DCN model introduces the Cross Network.The Cross Network consists of crossover layers, and the equation for each layer is as follows:

$$x_{k+1} = x_0 x_k^T \omega_k + b_k + x_k = f(x_k, w_k, b_k) + x_k \tag{11}$$

In the above formula, x0 is the output of formula (11) and the input of formula (10).X_k and X_{k+1} are column vectors, representing the output of the intersection layer of the l layer and the (K+1) layer, respectively. ω_k, b_k are the weight and bias parameters of the layer l layer.After completing a feature intersection f(x), each intersection layer adds the original input feature vector,In fact, after each cross-layer mapping, it corresponds to a residual X_{k+1}-X_k.

As the layer gets deeper, the order of the corresponding input feature will increase accordingly after passing through the Cross Network.For K layers, the highest polynomial of the input feature X_0 is of order K+1.In fact,

$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$

Cross Network consists of these cross terms , corresponding to orders from K to K+1.

Due to the low number of parameters of the Cross Network, the capacity of the model is limited. In order to obtain high-order nonlinear intersection features, a deep network can be added, and each Layer has the following equation :

$$h_{k+1} = f(\omega_k h_k + b_k) \tag{12}$$

Among them: $h_k \in R^{n_k}$ and $h_{k+1} \in R^{n_{k+1}}$ are the parameters of the hidden layer of the K layer and the K+1 layer, respectively, $w_k \in R^{n_{k+1} \times n_k}$ and $b_k \in R^{n_{k+1}}$ are the parameters of the K depth layer, and $f(x)$ is the ReLU function.

Finally, the output of the Cross Network and the output of the Deep Network are spliced together, and then the spliced result is used as the input feature of the logistic regression model, and the gradient descent algorithm is used for optimization. Finally, the sigmoid is used as the activation function of the output layer to output the result.The equation is as follows:

$$p = \sigma((x_{k1}^T, h_{k2}^T)\omega_{logits}) \tag{13}$$

In the above formula: p is the final prediction probability; x_{k1}^T represents the final output of Cross Network; x_{k2}^T represents the final output of Deep Network; ω_{logits} is the weight of Combination Layer.

The DCN model uses a cross network to learn high-order features using Feature Crossing at each layer. The characteristic is that the network structure is simple and efficient. Compared with DNN and DCN, the corresponding logloss value is lower, and the number of parameters is one order of magnitude less. However, the hyper-parameters in the Cross Network and Deep Network, including the number of layers of the crossed neural network and the number of neurons, cannot be set accurately; at the same time, if the dimension of the feature is large and the sparseness is serious, it will lead to the efficiency of the DCN model decline.

3.3 xDeepFM

xDeepFM[10] consists of three parts, namely DNN, Linear and CIN. After the embedding layer, the DNN will be connected with the CIN module and connected to the simple multi-layer perceptron in parallel. Linear is a simple linear regression model with raw features as input. The main function of CIN (compressed interaction network) is to learn explicit intersection features. As shown in Figure 3:

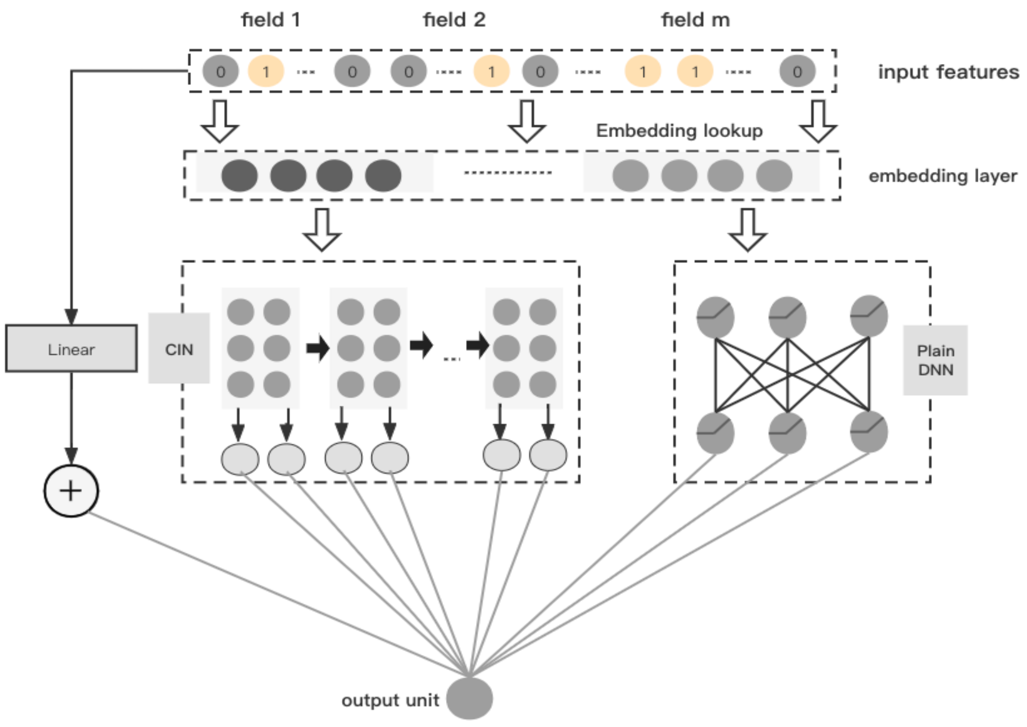

Figure 3: The Structure Of The xDeepFM Model

The purpose of CIN is to generate cross-features in a vector-wise manner in an explicit manner, as evidenced by it has some common properties with Convolutional Neural Networks[11] (CNN) and Recurrent Neural Networks[12] (RNN). The proposal of CIN solves the problem of feature intersection, and the order of intersecting features is the number of layers of CIN.CIN is a multi-layer output structure, each layer will train a matrix, the matrix represents the matrix trained by the K layer as $x^k \in H_k \times D$, let $H_0 = m$, where H_k represents the number of feature vectors of the K layer, the formula is as follows:

$$x_{h,*}^{k+1} = \sum_{i=1}^{H_k} \sum_{j=1}^{m} W_{i,j}^{k+1,h} (x_{i,*}^{k} \times x_{j,*}^{k}) \tag{14}$$

- $x_{h,*}^{k+1} \in R^D$ represents the feature vector of row h in X^{k+1} .
- $x_{i,*}^{k}$ represents the feature vector of the i-th row of the output matrix d of the k-th layer of CIN.
- $x_{j,*}^{k}$ represents the j-th row feature vector of the original feature matrix X_0.
- $W_{i,j}^{k+1,h} \in H_k \times m$ is a scalar value representing the parameter at row i and column j.
- $\times$ is an operator representing the Hadamard product, i.e. the inner product of matrices of the same order.

After the output of each layer is obtained, all feature matrices are compressed into a new vector through the sum pooling operation as the final output of CIN. The formula is as follows:

$$y = \frac{1}{1+e^{(p^{+})^{T}W^{o}}} \tag{15}$$

W^o is a parameter vector of length $\sum_{k=1}^{H^k}$ .The Linear, CIN, and DNN are unified together as the final output of the model, as shown in the following formula:

$$\hat{y} = \sigma\left(w_{linear}^{T}a + w_{dnn}^{T}x_{dnn}^{k} + w_{cin}^{T}p^{+} + b\right) \tag{16}$$

- a is the original input feature.
- x_{dnn}^{k} is the output vector of DNN.
- p^{+} is the output vector of CIN.
- w_{linear} , w_{dnn} , and w_{cin} are the training parameters of the corresponding modules, respectively.
- b is the global bias term; σ is the activation function.

3.3 DND-P

DND-P (Deep Model in Nacos Discovery Protocol) is a public package developed on the basis of the Python programming language and Django as a web framework. It can be installed directly through the pip command when building Django web applications. Among them, the DND-P algorithm is described as follows:

Algorithm 1 Calculate DND-P

```
Require: status is a flag having three value: 1-starting, 2-running and 0-stopping. Initial value is ‘1’.
Ensure: status = 1
1: global variable p
2: if status = 1 then
3:     url ⇐ ‘registering nacos server api’
4:     request_params ⇐ ‘register server’
5:     response_result ⇐ requests(url, request_params)
6:     if response_result.status = 200 and response_result.msg = ‘ok’ then
7:         p ⇐ new asynchronous process
8:         p.start()
9:     end if
10: else if status = 0 then
11:     url ⇐ ‘deleting nacos server api’
```

```
12:     request_params $\Leftarrow$ 'delete server'
13:     response_result $\Leftarrow$ requests(url, request_params)
14:     **if** response_result.status = 200 and response_result.msg = 'ok' **then**
15:       p.stop()
16:     **end if**
17: **else**
18:     url $\Leftarrow$ 'beating nacos server api'
19:     request_params $\Leftarrow$ 'beating server'
20:     current_time $\Leftarrow$ datetime.now()
21:     **while** p **do**:
22:         **synchronize**({
23:             one_minute = datetime.now()
24:             **if** one_minute - current_time = 60 seconds **then**:
25:                 requests(url, request_params)
26:                 current_time = one_minute
27:             **end if**
28:         })
29: **end if**
```

3 Experiment

3.3 DataSet

The experiment uses the real data set published by Criteo. Criteo is a classic data set in the CTR field. The features are desensitized. There are 13 continuous features and 26 category features. The samples are sorted by time. This experiment randomly sampled 600,000 sample sets, of which the number of positive samples is: 153,822, and the number of negative samples is: 446,178.

3.3 Environment

The environment for training DCN, DeepFM, and xDeepFM models is shown in Table 1:

Table 1: Deep Learning CTR Model Environment

Type Of Environment	Configuration Parameters	
Hardware Environment	Host Model	Aliyun ecs.vgn5i
	CPU	4 core(vCPU)
	GPU	Tesla P4*1/4
	Memory	24 GiB
	Hard Disk	SSD cloud disk 40GiB (2400 IOPS)
Software Environment	Operating System	Ubuntu 20.04.4
	Development Language	Python 3.8
	Deep Learning Framework	tensorflow-gpu 2.2.0

The environment of the Microservice Architecture is shown in Table 2:

Table 2 Microservice Architecture Environment

Type Of Environment	Configuration Parameters	
Hardware Environment	Host Model	Aliyun ecs.c5.large
	CPU	4 core(vCPU)
	Memory	8 GiB
	Hard Disk	SSD cloud disk 40GiB (2120 IOPS)
Software Environment	Operating System	Ubuntu 18.04
	Development Language	Python 3.8, Java 8
	Microservice Framework	Spring Cloud Hoxton.SR9 Spring Cloud Alibaba 2.1.0.RELEASE Nacos 1.1.4 Spring Boot 2.3.7.RELEASE
	Web Framework	Django 2.0

In the data set, the positive/negative samples are divided by 8:2, and then combined, so that the overall allocation is 80% as the training set and 20% as the test set.After many experiments, the optimal parameter combination and model prediction results are obtained, as shown in the following table 3:

Table 3 Experimental Results

Model	Hyper-parameters							Evaluation Metrics	
	batch size	embedding	Dropout	CIN Layers	CIN Output Filters	Cross Net Layers	epochs	AUC	Loss
DCN	256	128	0.1	-	-	3	5	0.7330	0.5407
DeepFM				-	-	-		0.7404	0.5723
xDeepFM				3	12	-		0.7402	0.5756

The optimal combination obtained after many experiments is that the batch size is 256; the embedding layer of DNN is 128 layers, and the number of neurons is 128; the L2 regularization penalty coefficient is 0.7, the dropout is 0.1, and the learning rate is 0.001; The CIN network has 3 layers and the output dimension of each layer is 12; the Cross Network is 3; the activation function is ReLU; the number of iterations is 5.

4 Conclusion

First create a namespace of ctr_deep_model in Nacos. Then, start the DCN, DeepFM, and xDeepFM Web Servers that integrate the DND-P protocol respectively. The corresponding service names are ctr_dcn_model, ctr_deep_fm_mode, and ctr_x_deep_fm_model. Each service actually represents a cluster, and there is one instance in each cluster. Finally, the microservices developed in Java are called, and 5 positive samples and 5 negative samples are randomly selected from the 20% test set for testing, and finally good results are obtained.

It can be concluded that the DND-P protocol can quickly realize the integration of the microservice architecture and the deep model in the cloud platform, and it can integrate well in different languages and under different frameworks.

References

[1] Sun Bing. Analysis of application architecture and development trend of cloud native in enterprises[C]//.Proceedings of the 25th Annual Conference on New Network Technologies and Applications in 2021 by the Network Application Branch of China Computer Users Association.,2021:412-418.DOI:10.26914/c.cnkihy.2021.047868.

[2] Alibaba Group, Alibaba Cloud Intelligent Business Group, cloud native application platform.Alibaba Cloud Cloud Native Architecture Practice[M]. Beijing: Machinery Industry Press. 2021.5..

[3] XIN Yuanyuan, NIU Jun, XIE Zhijun, et al. Survey of implementation framework for microservices architecture.Computer Engineering and Applications, 2018, 54（19）：10-17.

[4] Feng Zhiyong，Xu Yanwei，Xue Xiao，and Cheng Shizhan.Review on the Development of Microservice Architecture.Journal of Computer Research and Development.,DOI: 10.7544/issn1000-1239.2020.20190460 57(5):1103-1122, 2020.

[5] https://github.com/alibaba/nacos.

[6] https://www.djangoproject.com/start/overview/.

[7] Guo H, Tang R, Ye Y, et al. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction[C]// Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017.

[8] Wang R, Fu B, Fu G, et al. Deep & Cross Network for Ad Click Predictions[C]// ADKDD'17. ACM, 2017.

[9] Cheng H T, Koc L, Harmsen J, et al. Wide & Deep Learning for Recommender Systems[C]// Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016.

[10] Lian J, Zhou X, Zhang F, et al. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems[C]// the 24th ACM SIGKDD International Conference. ACM, 2018.

[11] Qiang L, Feng Y, Wu S, et al. A Convolutional Click Prediction Model[C]// Acm International on Conference on Information & Knowledge Management. ACM, 2015.

[12] Zhang Y, Dai H, Chang X, et al. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks[C]// Twenty-eighth Aaai Conference on Artificial Intelligence. AAAI Press, 2014.