

CTR-Prediction Model Architecture for Large-Scale Distributed Machine Learning

Yan Dai

Reed-AI, Beijing, China

yandai@reed-ai.com

ABSTRACT

We have designed a CTR&CVR prediction model architecture based on large-scale distributed machine learning and deep learning. This paper can be roughly divided into two parts: content delivery and content generation. Content delivery, also known as information flow delivery, is applied to search, recommendation, advertising content traffic delivery platforms to complete content distribution and delivery. The other part is content generation, which is a generative large model based on AIGC multimodal technology. The first part of the article is an overall introduction to the work. The second part introduces the previous industrial work and the latest developments in this field. The third part mainly covers the engineering infrastructure, including offline data warehouses and real-time data warehouses near the line, distributed databases, distributed model training frameworks, online micro-service architecture, as well as performance optimization methods and final optimized system performance indicators. The fourth part is the design of algorithm principles, which improves and comprehensively applies mainstream models in the industry, and proposes a new algorithm model. The fifth part compares and verifies the algorithm model through experiments, and records the performance of the algorithm experiment on different measured datasets. In the sixth part, based on the above work, we propose a large-scale distributed machine learning and deep learning model training framework, which is applied to multi cluster parallel training to improve model training efficiency and resource utilization. The seventh part briefly introduces the final conclusion of the algorithm work, and the current algorithm design has better performance than mainstream models.

KEYWORDS

CTR-Prediction, Large-Scale Distributed, Recommender System

1 INTRODUCTION

In recent years, the information flow delivery technology in the field of Internet ToC users has become increasingly perfect. The search, recommendation and advertising technology based on machine learning and deep learning has developed into a mature industrial chain

ecosystem, providing continuous business growth for the delivery platform. At the same time, with the popularization of Internet information technology, the industry has gradually changed from the initial explosive user growth to the stock user mining in specialized fields. At the same time, the technology applications for B2B merchants in traditional industries are gradually showing their potential for broad application space.

Our main job is to first create a traffic delivery platform for search/recommendation/advertising based on machine learning and deep learning, and complete the distribution and delivery of content information. Mainly based on big data technology and personalized content recommendation, it includes different systems such as offline, near line, and online services. Through targeted traffic distribution to different users, it can improve user click through rates and activity, enhance user retention on the platform, and also improve user conversion rates through specific optimization of personalized recommendation technology. Secondly, the content generation section. At the same time as the distribution and delivery of content information flow, due to the limited number of recalled content materials, which are all manually obtained from real video recordings or live content, the content production efficiency is too low, resulting in a large number of users still unable to receive effective and interesting content recommendations in rich behavioral scenarios. Therefore, we have designed a content generation algorithm that is integrated into the model, using AIGC multimodal technology that integrates images, videos, speech, and text. Based on a large model trained generative large model, it eliminates the manual annotation of video data in traditional supervised learning, saves a lot of manual operation steps, makes the delivery content more abundant, provides users with better personalized and diversified experience, improves user retention, and increases platform revenue.

In addition, we propose a new distributed model training framework to meet real-time model training and updates under given data volume and established high traffic. This distributed parallel model computing framework can more efficiently complete model training, solve the efficiency bottleneck of existing algorithms, and has better scalability and universality. We have launched 6 versions of experiments and iterations before and after, and completed

full iteration replacement. Currently, CTR and CVR have caught up with the manual team. By supplementing some global fallback logic, we have covered different types of users, including new users/active users/inactive users/pre churn users. Realize comprehensive improvement in click through rate and conversion rate.

Finally, we integrated this series of algorithm work into the computing platform, forming machine learning and deep learning platforms as the infrastructure for subsequent business development, enabling better and faster observation and iteration of experimental results. At the same time, in order to solve the problem of large-scale distributed model training under high traffic, we have made detailed splits and upgrades for data parallelism and model parallelism during the model training process.

2 RELATED WORK

YouTube-DNN^[1], for video information flow recommendation, a multi classification model is used for the recall part, with the optimization objective of the next recommended video. The negative sample is the exposed but unfinished video, and the weight is the video playback duration, which is equivalent to weighting the positive sample. This weight will directly act on the loss function, namely cross entropy. The model training label is video ID, which is a million level multi classification model training task. The input features have a total of about 1000 dimensions, including: video id embeddings of videos watched by users, about 200 dimensions; The embedding of the user search term token id is approximately 200 dimensions; Unilateral features such as user behavior and attributes, approximately 300 dimensions; The unilateral characteristics of the material, including video popularity and freshness, are approximately 300 dimensions. In the later stage, the model training is accelerated by 100 times through negative sampling of the samples. The overall model still uses offline calculation of item feature embeddings and online real-time calculation of user behavior feature embeddings to obtain similar vector recalls. Specifically, similarity calculation is used, that is, the nearest neighbor search method is used to calculate the vector cosine, and then retrieval is performed by truncating the topN recall item set according to the sorting score. The precision ranking model used in the sorting section adopts a binary classification scoring model, with positive samples being videos exposed and clicked, and training weights being video playback duration.

DSSM^[2], uses a deep learning model that calculates semantic similarity to solve the problem of traditional keyword matching models being unable to capture semantic

similarity. The core idea is to map the query statement and document doc to the same low dimensional semantic space, and calculate the vector distance between the two through cosine similarity to evaluate the degree of semantic matching. The model structure adopts a dual tower structure, including a user tower and a material tower. The query and document are converted into low dimensional vectors through deep neural networks, and the output dimensions of the two towers need to be consistent to ensure feature alignment. The specific application scenarios are mainly used in the recall stage of search scenarios, such as matching the semantic similarity between queries and documents in search engines, and predicting the degree of interest matching of users for materials in recommendation systems. Its advantage lies in handling subtle language differences and can be extended to other similarity calculation scenarios. Its advantage is that it can effectively handle semantic matching and generate sentence embedding vectors for similarity prediction. At the same time, the twin tower structure cannot capture the interaction information of features on both sides, which may affect accuracy. Moreover, model training relies heavily on a large amount of click or exposure log data.

Deep interest network (**DIN^[3]**) is mainly used to solve the problem that traditional recommendation systems cannot effectively capture users' diverse interests. Its core is to dynamically generate user interest representations through attention mechanisms to improve recommendation accuracy. The core idea is to dynamically generate user interest representations by assigning weights to different products through attention mechanisms. Unlike traditional models that use fixed vectors to represent users, DIN dynamically adjusts interest weights based on user historical behavior and the relevance of candidate products. For example, if a user has browsed a certain product, their interest in related products is higher. The overall model structure is divided into an embedding layer, which converts user history behavior and candidate products into low dimensional vectors. The attention calculation layer generates difference vectors through operations such as element cutting and concatenation, and inputs them into the fully connected layer to obtain attention scores through the softmax function. Interest interaction layer, combined with user historical behavior and current product features, extracts potential features through multi-layer perceptron (MLP). Its advantages can be summarized into the following two aspects: firstly, dynamic interest expression. Dynamically adjust user interest weights based on different products to avoid single fixed vector restrictions. Secondly, it has a wide range of application scenarios and is suitable for fields such

as e-commerce, news, music, etc. Especially in e-commerce scenarios, it optimizes recommendation effects through user historical behavior, such as browsing and purchasing records. It has achieved significant results in practical scenarios such as large-scale advertising platforms and has become one of the mainstream recommendation models in the industry.

The click conversion estimation model **ESMM**^[6] is a multi task learning model, but mainly focuses on the estimation of click through rate (CTR) and conversion rate (CVR). It is mainly used to solve the problems of sample selection bias (SSB) and data sparsity (DS) in post click conversion rate (CVR) estimation in recommendation systems. The core idea is to model the entire behavior sequence space of users, share feature expressions, and jointly train two tasks: click through rate (CTR) and post click conversion rate (CTCVR). Its core lies in alleviating the problem of data sparsity and eliminating the impact of sample selection bias through multi task collaborative training. Its usage scenarios are mostly e-commerce, information retrieval, online advertising, and other scenarios that require accurate estimation of conversion rate (CVR). For example, in e-commerce platforms, by estimating the purchase probability (CVR) of a product after being clicked, optimizing the ranking of product recommendations, and maximizing the total transaction volume (GMV). The main advantages of this model are threefold: firstly, it solves the problem of sample selection bias (SSB). Traditional CVR models only use click samples for training, while ESMM models in the entire sample space, avoiding the problem of inconsistent training and actual predicted distributions. Secondly, to alleviate the problem of data sparsity (DS), the CVR model can be trained using shared feature expressions and non clicked samples to reduce the issue of data sparsity. Thirdly, multi task collaboration, while optimizing CTR and CTCVR, enhances the model's generalization ability.

Multi expert model **MMoE**^[7], a general multi task learning model, solves the problem of poor performance of traditional multi task models when there are significant differences in tasks by introducing a mixed expert structure. The core idea is to use gate controlled networks to assign weights to different tasks, thereby flexibly combining underlying shared features. MMoE consists of three parts: sharing the underlying layer, extracting common features of input data; The expert layer consists of multiple expert networks (MoEs), each expert processing features from different dimensions; Gated network generates allocation strategies for each task and dynamically adjusts the utilization of underlying features based on task requirements. The core advantage of its model lies in: firstly,

avoiding parameter inflation, dynamically combining shared features through gate controlled networks, and adapting to different tasks without adding additional parameters; Secondly, flexibly capturing differences, displaying the correlation and differentiation between learning tasks, and improving the learning efficiency of each task; Thirdly, it has a wide range of applications, including recommendation systems for estimating click through rates and conversion rates, search engines for estimating relevance and satisfaction, and more commonly used in multi-objective optimization scenarios. It has good universality for linear and nonlinear multi-objective combinations that cannot be determined or dynamically transformed. Compared with traditional multi task learning models, such as sharing underlying structures, MMoE forces all tasks to share underlying parameters, which can easily lead to overfitting or performance degradation when there are significant differences in tasks. MMoE alleviates this issue through gating mechanisms, but requires more data training.

DeepSpeed, a distributed training framework and deep learning optimization library, focuses on improving the efficiency and scalability of large-scale model training. It uses multiple core technologies to reduce video memory usage, accelerate the training process, and seamlessly integrates with mainstream frameworks such as PyTorch, supporting distributed training and training of multi billion level parameter models. Its core function is mainly: firstly, video memory optimization, which uses ZeRO series technology to shard model states to multiple GPUs, achieving a linear decrease in video memory usage with the number of GPUs and supporting larger model training. Secondly, communication optimization, the 1-bit Adam algorithm compresses communication data, reducing communication volume by 5 times and improving training efficiency. Thirdly, mixed precision training automatically switches between semi precision floating-point numbers (FP16) and full precision floating-point numbers (FP32), balancing precision and computational efficiency. Its framework has good compatibility, adapts to various communication strategies such as MPI and MCCL, and is compatible with consumer grade graphics cards (such as T4, 3090) and cloud server graphics cards; At the same time, it has good scalability and supports distributed training from single machine multi card to kilocard level clusters, suitable for language modeling, image processing and other scenarios. At present, it has been widely used for training large-scale language models such as GPT-3 and Turing NLG, and continues to iterate and optimize.

3 INFRA ARCHITECTURE

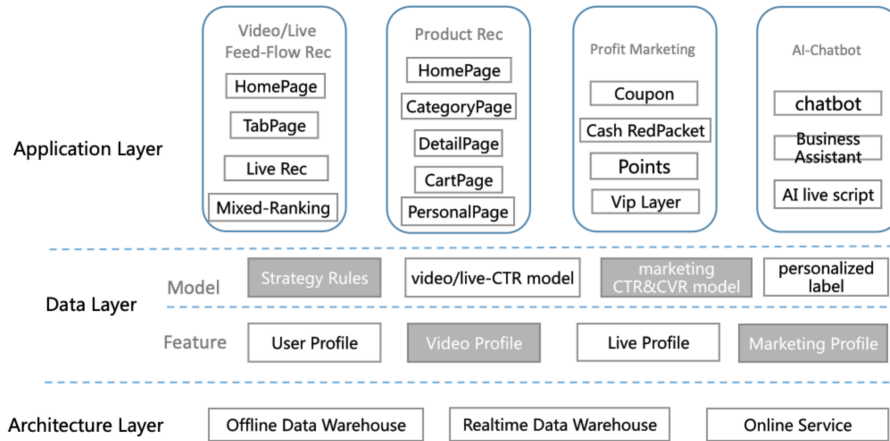


Figure 1: Business Architecture

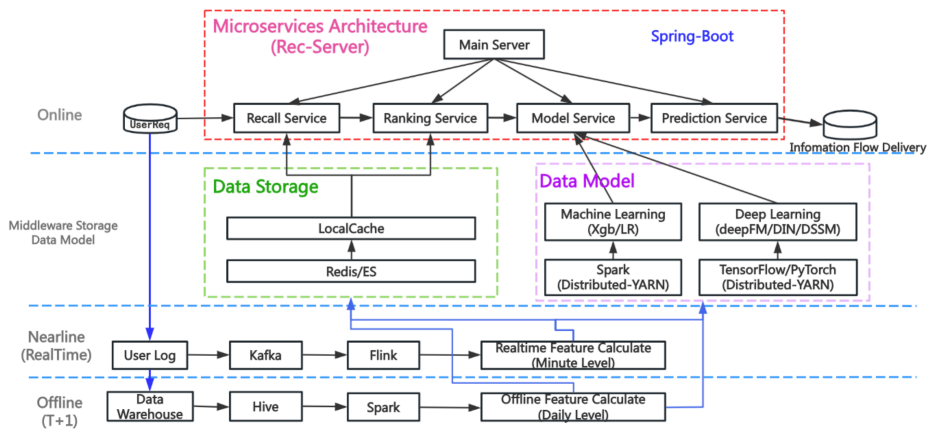


Figure 2: Engineering Architecture

3.1 Offline Architecture

In the monitoring log of the buried point, all user behavior data is recorded in the original log and stored in the database of the online service by block size. By parsing data of different time granularities into the data warehouse, the original form of user behavior actions is preserved. Based on logging through Kafka message queues, real-time user information streams can be buffered and recorded. Then, the Flume distributed logging framework can capture user actions and behaviors from different online servers and store them in offline HDFS. It is necessary to ensure the timestamp order on different distributed machines, achieve synchronous recording, and ultimately aggregate it into an offline data pool. Data relies on general persistent storage media, structured data flows into PolarDB or other storage media based on MySQL, and unstructured data generally flows into MongoDB to preserve the original data features. The data extracted from the database, after standard formatting, is stored in Hive in a structured form. Typically, the offline data warehouse updates all data at a T+1 time

frequency. The offline data warehouse is divided into five layers from bottom to top based on different storage forms: the operational data layer (ODS layer), which stores raw data with a structure similar to the source system and serves as a data preparation area; The data warehouse detail layer (DWD layer) constructs the most fine-grained detail data tables, designs wide tables based on business process characteristics, and associates dimension model primary keys with fact table foreign keys; The data warehouse summary layer (DWS layer) constructs a public indicator summary table based on topic objects, uses a wide table approach to statistically measure caliber, and provides support for upper level application of public indicators; Dimension layer (DIM layer), with dimension modeling as the core, defines dimension attributes and associated logic, and constructs the dimension table of the snowflake model; The application data storage layer (ADS layer) stores the final application data and exports it to databases such as OLAP and RDS for business use.

Offline feature engineering mainly involves offline overall pipeline processing, sinking into the feature

middleware. Converting raw data into machine learning model inputs is mainly achieved through batch data processing, suitable for scenarios of data consistency and offline computing. Generally, the main parts of feature engineering include: firstly, data collection and cleaning, integrating offline data such as user behavior, item attributes, and cross behavior data, and handling missing values and abnormal data; Secondly, feature construction involves generating new features through user profiles, item classifications, and other methods, such as the user's recent activity time, user click browsing behavior, user basic attributes, and material category distribution; Thirdly, feature processing involves data discretization, normalization, pooling, and other operations to improve data quality, such as converting continuous features into piecewise functions; Fourth, evaluate optimization by verifying the effectiveness of features through offline models and selecting significant features that improve business indicators such as click through rates and conversion rates. The processing cycle of offline feature engineering is relatively long, usually measured in days or hours, with a focus on batch correction and long-term effect optimization. In general application scenarios (such as e-commerce recommendation systems), offline features are often used to modify order delay feedback and calculate users' long-term interest preferences.

Offline cluster, a distributed computing system that processes massive historical data. Built on Hadoop/Hive, data cleaning, aggregation, and analysis tasks are completed through batch processing, supporting large-scale data storage and complex computing, and carrying functions such as data storage, data processing, and data analysis. The data is stored in blocks using distributed storage (such as HDFS) to ensure high availability and reliability; Batch processing of data through MapReduce framework for data cleaning; Utilize the Spark computing framework for statistical analysis and machine learning model training. The overall technical architecture follows several principles, including offline data warehouse layered design, resource scheduling through Yarn and other components, dynamic allocation of CPU and memory resources, and support for elastic expansion; Normally, offline clusters do not require a second level response, but rather complete data processing during idle time periods, reducing real-time requirements in exchange for higher throughput and cost optimization.

3.2 Nearline Architecture

Nearline distributed cluster, real-time information flow log processing, realizing data second level updates and queries. For buried point logs within 24 hours, stream

feature calculation and stream model are used to generate a near line estimation model. The key function is to supplement real-time user behavior data and real-time detection of newly registered users, with a time frequency generally at the minute and second level. The main features are distributed transaction processing, high concurrency support, and data consistency assurance. From bottom to top, it is divided into three layers: the Operational Data Storage layer (ODS layer), which stores temporary raw data and generally collects data using Kafka message queues. Its underlying metadata is saved through Zookeeper; The real-time data warehouse core layer (EDW layer), the real-time computing engine Flink streams data, which can usually be divided into batch processing and stream processing. The most commonly used method is stream processing to record data, which relies on yarn resource management to execute tasks. By reading Kafka stream data in real time to data storage such as Hive, real-time data storage is achieved; Application data storage (ADS layer), data view for business scenarios, supporting timely analysis and online data services. Compared with traditional offline data warehouses, there are more scenarios where data is promptly responded to and TB level data incremental updates are processed.

Online feature engineering is mainly used for using near line models to supplement real-time user behavior features and real-time CTR estimation. Using 90 days offline as the baseline, incremental training will be conducted daily, and online data will be collected every 10-15 minutes for model training. The data is retrieved from Kafka using Flink and dropped onto Hive. At each batch processing time node, it is carried forward and the online feature weights are overwritten. The online estimation model uses FTRL as the baseline for the incremental model, while optimizing the model training speed. For online models, the release cycle can be shortened to 1 day and updated every 1 hour.

3.3 Distributed Model Training

Distributed model training is mainly divided into two directions: data parallelism and model parallelism.

(1) Data parallelism: mainly targeting the standalone version of Python, the training sample data is sharded, mainly for the horizontal splitting of the training set, ensuring that the number of training samples in a single shard is within a certain range, so that the training set data and the model.DataFrame generated during the training process are less than the maximum memory. Usually, by combining feature dimensions, the training set matrix does not exceed the memory size, and finally the shard prediction results are aggregated and concatenated to obtain the final

model inference result. Because the Python version of the model training still belongs to a standalone model, it relies on the basic matrix structure of `Pandas.DataFrame`. Each time a task is submitted to the cluster during training, all data will be loaded into the computing container. This basic data unit cannot be split, so the minimum data block size depends on the size of the training set after data fragmentation.

(2) Model parallelism has improved the training of Python standalone models. Based on the existing Python based parallel matrix `dask` and parallel library `Ray`, parallel computing can be directly performed on standalone nodes; In addition, by directly relying on the distributed computing framework, distributed Spark XGB, the machine learning library of Spark, an existing distributed computing framework in the industry, can directly perform distributed computing on common machine learning models on some appropriate basis. The basic mechanism is to drive the Driver node to submit job tasks to the distributed Worker sub nodes for direct distribution to each computing node to complete the computation. This method can indeed effectively improve the computational efficiency of model training within a certain range of traffic levels, but is limited by basic parameters such as the given number of computing cluster nodes, maximum memory value of computing server nodes, and server frequency. In some business scenarios that do not require high real-time computing, it can indeed effectively alleviate the bottleneck problem of computing, but is limited by the number of established algorithm model libraries. We have made improvements to the `DataFrame` part of Spark's basic data unit and optimized the scheduling scheme.

3.4 Online Service Architecture

Data storage stores retrieval user IDs and recall content result sets. Structured data is stored in MySQL, but is usually stored in Redis in the form of KV to achieve high-speed read and write, caching of data, and session management. It is based on memory data structure as the storage system, and atomic operations ensure data consistency. It supports two persistence methods, snapshot and file append, to ensure data security and achieve high availability. At the same time, ES retrieval has been introduced in scenarios that require fast full-text retrieval, applied to various combination condition queries. The distributed architecture supports horizontal expansion, multi node processing of massive data, and achieves near real-time search. Data can be retrieved immediately after writing, and each field can be indexed. Add a local cache between the database and online services, where data is stored in

memory and can be quickly accessed locally. This improves program efficiency, reduces the frequency of accessing backend data elements, lowers backend load, and enhances overall performance. To improve the overall throughput of the system, the single machine is expanded to a combination of distributed and multi-source databases, forming a distributed database cluster, effectively alleviating query load pressure, and increasing the overall QPS by 300%.

Distributed online micro-services: Information flow delivery retrieves recall data from different data sources through online interfaces for post-processing. The overall steps are independent of each other, with only interface dependencies, involving the main parts of recall, sorting, modeling, and estimation. The application is divided into multiple small, autonomous service units, deployed and maintained separately, and data is exchanged between them through lightweight communication interfaces. Each service focuses on a single business function, has an independent database and code, supports loose coupling architecture and elastic expansion. Ensure overall data consistency, and each service's technical stack exists independently and separately. Information flow delivery covers interfaces across multiple business lines, such as video/live streaming, online shopping, advertising, and equity activity delivery.

As shown in Figure 2, the user's online request will be parsed to reach the recall source corresponding to the matching response feature in the database. The corresponding recall data will be extracted from the recall service, and then the sorting service will rank all the recall source sets based on the ranking score. The model service will then extract the corresponding model calculation score, and finally enter the estimation service to obtain the material sorting results of the online CTR and CVR estimated scores. Finally, the results will be presented to the user. All online micro-services retrieve data from cache or database, and model services retrieve pre updated model results from model results. The data sources for databases or models are offline data warehouses and real-time data warehouses, and the data sources for these two parts of the data warehouse are online user behavior logs, which are recorded in real-time or daily in the logging system and then processed into offline tables for the data warehouse. Thus forming a closed-loop pathway online and offline.

Stress testing is conducted on online services using `Wrk`. The parameters are: `t` is the number of simulated threads, `c` is the number of simulated connections, `t` is the timeout, and `d` is the testing time. For example, 'wrk-t2000-d30s - T5s' represents 4 threads and 2000 connections. Within 30 seconds, the stress testing time interval is 5 seconds, and the latency parameter shows the distribution of

response time. In the test results, Latency is the response time, Req/Sec (QPS) is the number of completed requests per second for each thread, Avg reflects the average size and Max maximum value, and Stdev standard deviation represents the proportion of plus or minus one standard deviation. If the standard deviation is too large, it indicates that the sample itself has a high degree of dispersion It is

	t4-C20	t4-C30	t4-C40
Redis	TTS: 589.68ms Requests/sec: 56.97	TTS: 540.65ms Requests/sec: 57.80	TTS: 794.92ms Requests/sec: 57.69
ES	TTS: 529.72ms Requests/sec: 61.86	TTS: 701.66ms Requests/sec: 57.64	TTS: 2.00s Requests/sec: 226.19

Table 1: Comparison of Service Stress Test Results

Monitor and access Prometheus services to analyze monitoring logs of important interfaces on online servers, and display monitoring results using the Grafana control panel to ensure timely tracking of data results.

- (1) Service monitoring: interfaces and important traffic bits;
- (2) Offline data and scheduling task monitoring: data correctness and accuracy;
- (3) Log rolling segmentation, log embedding and storage, based on Kafka, Flume, and Hdfs;
- (4) Service architecture and operations: monitoring alarms, service disaster recovery, high availability;
- (5) Data dashboard: Develop business indicator monitoring for each traffic level based on Superset.

3.5 Performance Optimization

We have optimized various parts of the service performance:

- (1) Operator optimization: recall, sorting, modeling, estimation, micro-service splitting optimization;
- (2) Service expansion: Nginx offloading and Zookeeper load balancing are adopted, while AB offloading configuration is implemented. Each specific user request will be allocated to different traffic experiments based on the ID number for comparative analysis of results; The service has added Jdk performance analysis, utilizing jviewvm, jstack, jstat, including code blocks, program stack, and application block occupancy time, to optimize program execution efficiency and memory usage space respectively;
- (3) Code quality: execution time optimization, memory usage optimization, operator module decoupling optimization, operator instruction set optimization;
- (4) Data query time: query IO optimization, database optimization, distributed optimization, index optimization;
- (5) Cache hits: connection pool optimization, index optimization.

3.6 Engineering Results

possible that the system performance fluctuates greatly. The number of threads used for simulation testing is approximately 2-4 times the number of machine cores, which should not be too many. Otherwise, efficiency may decrease due to excessive thread switching. WRK does not use a model of one thread per connection, but increases concurrency through asynchronous network IO.

After optimization, the online service has a system throughput QPS of 71 for a single machine before optimization and 136 for a pre release machine after optimization. After upgrading the distributed cluster, the estimated total throughput of all clusters is $136 \times 6 = 816$, resulting in an overall performance improvement of 11.5 times and an improvement in TP99. The TTS full link response time was 600~700ms for a single machine before optimization, but after optimization, the response time for a single machine was within 100ms, resulting in an overall improvement of 7 times.

In terms of model training, Python standalone version was initially used to shard the training set data, with a single shard training time of 1 hour and 44 minutes. After optimization, the training time was shortened to 30 minutes, and the overall training time was increased by 300%; Then, using the Spark XGB distributed version, the training time was optimized from 30 minutes to 10 minutes, resulting in a 300% increase in efficiency.

The final optimized QPS estimation includes an information flow delivery system that includes online clusters, offline distributed clusters, and distributed databases. The number of users that can be supported has increased from 300w of small traffic to over 1000w of total traffic, meeting the full capacity standard.

4 ALGORITHM DESIGN

4.1 Model Architecture Variants

For the personalized recall part, we have improved the model YouTube DNN^[1], which was originally trained as a multi class model. In the current video streaming recommendation scenario, we have re modeled user characteristics including: basic user attributes, user browsing, clicks, valid clicks, forwarding, commenting, liking, sharing, bookmarking, liking, user geographic location, user purchasing power, user order related videos, and added negative feedback features such as unliking and

unbook-marking. The characteristics of video materials include: the number of clicks, shares, comments, likes, shares, bookmarks, likes, video start rate, video completion rate, video playback duration, whether the video has order product tags, video popularity, video freshness, and vector recall. It is important to differentiate between sparse and dense features and embed them separately for modeling. The label granularity of overall multi class training is used for shallow clustering, as shown in the DHAN^[5] improved model below.

For the recall matching part, we have improved the model DSSM^[2], offline feature training, online vectorized similarity recall inference and estimation. The overall recall is divided into N towers based on different businesses, including user towers, product towers, copywriting towers, and video towers. After embedding the multi-dimensional features of the user, a DNN multi-layer neural network is trained and then mapped to a low dimensional vector space; Similarly, after embedding the characteristics of other material towers, the processing remains as a multi-layer neural network. Based on word2vector, the dense and sparse features need to be separately embedded and hashed after being divided into buckets. Each feature learns a vector separately. When calculating vector similarity, filtering calculation is used to reduce the complexity of online recall calculation, while ensuring data compression to the normalized interval and ensuring data consistency.

4.2 Interest Network Variants

The improvement of user feature vectors, drawing on the ideas of DIN^[3], involves incorporating user behavior feature vectors, adding attention mechanisms, and generating dynamic interest vectors. The material content recently clicked and browsed by the user will be weighted, and different feature vectors will be integrated to calculate the comprehensive attention score, which will be input into the subsequent network for model training.

Due to the controversial nature of recalling multi class modeling granularity control, its focus needs to be determined based on specific business forms to determine the granularity segmentation range. In the case of massive data samples, the most extreme approach is to have each sample as a learning classification target. However, when the sample size reaches millions or tens of millions, this approach is obviously too computationally intensive for model training and is not practical in actual engineering. So it is necessary to model a multi class granularity model, following the approach of DHAN^[5], where diverse learning is aggregated into M-cid. The magnitude of this M is controllable and usually depends on the complexity of the

business. Then, the actual recall model learns an M-class model. At the same time, the sample must have orthogonality or at least ensure weak correlation among the aggregated M feature vectors, so that the model can learn better classification characteristics.

4.3 Sorting Algorithm

The sorting algorithm still adopts binary classification sorting, and information flow recommendation usually focuses more on activity, which is specifically manifested as CTR click through rate related indicators and video playback time. The sorting model adopts L2R model sorting to perform mixed sorting on multiple recalls, considering the sorting score of recall material relevance. The single-sided feature of the video is its own label attribute, and the similarity of retrieval IDs is used to determine the global importance of the video in the entire sorting list. The overall sorting list is optimized based on CTR and playback time.

After optimization and improvement, the overall CTR estimation model was used for sorting, and we optimized v1, v2, v3 and other three versions before and after. Version 1 directly models binary classification based on clicks or plays, and the generalization of the sorting results in applications is relatively weak. Version 2 iteratively optimized the selection of inter sample partitioning. Positive and negative samples are not in the global sample space, but are taken as click play and click unplay. At the same time, when downsampling a large number of negative samples, attention should be paid to the ratio. Version 3 extracts multi granularity modeling from the global sample space, paying attention to the weak correlation between granularities. The resulting ranking model has good click through rate performance.

4.4 Multi objective prediction

The overall business scope of information flow delivery still belongs to CTR&CVR modeling, and generally requires simultaneous optimization of click through rate (CTR) and conversion rate (CVR), which belongs to multi-objective simultaneous optimization. However, for this type of business with a small number of indicators and linear or nonlinear relationships that can be represented by functions, it is possible to try to find the functional relationships between indicators, split the optimization objectives, and optimize individual objectives separately, in order to achieve the ultimate optimization effect on the overall goal. For this special scenario, especially for the selection of sample space, the idea of ESMM^[6] was referred to. Before estimating the CVR, the click through rate CTR and post click conversion rate CTCVR were estimated to achieve

effective estimation of the final CVR. At the same time, attention should be paid to the sample selection for modeling the entire behavior sequence space of the user.

However, when the situation of multi-objective prediction becomes more complex, in most cross domain business scenarios, optimizing business indicators becomes unmodeled, that is, simple linear/nonlinear, proportional/inverse relationships cannot be used to describe the relationship between indicators, and even indicators are dynamically changing and influencing each other with sample data. At this time, even approximating the overall optimization objective function through linear representation becomes particularly difficult. A universal multi task learning model is needed, drawing on the ideas of MMoE^[7], to implement multi expert network inference, parallel large-scale experts, and directly bypass the problem of modeling the indicator relationship of the objective function. For the combination gate controlled network, the multi expert system adopts MSE multi task, multi expert writing, for example, 8 gates, which are experts from different business fields. Each context model is trained, and the output is the result of the second to last layer, paying attention to the broadcast of different layers. Multi task learning, expert system reasoning. In the later stage, adaptive learning is added, and the modeling playback time is no longer the main factor.

5 EXPERIMENTS

We divided the entire traffic into 6 groups and conducted experiments on them, using PolicyRules, U2i2i, U2u2i, Cid Pop, Als, CF, and an improved model DHAN^[5] for experimentation. Each bucket experiment is not a completely independent model, usually a combination with a fallback strategy. However, in order to verify the effectiveness of a single improved model as a control group, we strive to ensure the purity of the model as much as

possible. The final experimental grouping is scaled up to the maximum traffic, depending on the performance of the experimental indicators of the group, which are usually associated with business indicators such as CTR click through rate and CVR conversion rate.

The evaluation criteria for model experiments vary slightly depending on different business lines. For businesses that attract and promote user activity, such as video and live streaming, the evaluation indicators focus on the CTR click through rate category, which is subdivided into click through rate, effective click through rate, 50% playback progress, completion rate, and average playback time per person; For businesses such as promoting transaction volume and order conversion, the evaluation indicators focus on the CVR click through rate category, which is subdivided into user order conversion rate, revenue GMV, and cost ROI, such as the homepage, shopping cart page, and order page of the mall. In addition, it also includes equity businesses such as red envelopes, coupons, and points. The ultimate goal of all businesses is user retention and conversion, so it is usually necessary to jointly optimize various indicators, that is, the comprehensive indicator is CTCVR+ROI/GMV/CVR joint prediction.

The final evaluation results show that compared to the manual rule strategy, the comprehensive algorithm indicators are as follows: click through rate of 57.8%, which is better than the manual group's 54.8%; The effective click through rate is 24.6%, which is lower than the manual group's 31.5%. The average playback time per person is 22-24 minutes, which is better than the manual group's 19 minutes; The overall conversion rate is 27%, which is better than the manual group's 24%. The detailed granularity indicator data and trend chart are summarized in the following table:

GROUPS	CTR	Effective CTR	50% Playback Rate	Play-Through Rate	Per Capita Playtime
Algorithm group	57.858%	24.66%	9.901%	7.31%	22.50015
Control Group	60.084%	30.052%	21.413%	18.338%	24.25459
Artificial Group	54.884%	31.516%	18.666%	14.993%	19.80011

Table 2: Summary and Comparison Details of Click through Rates

GROUPS	CTR	Effective CTR	50% Playback Rate	Play-Through Rate	Per Capita Playtime
Algo1	55.761%	29.382%	18.914%	14.9%	13.81273
Algo2	53.378%	21.57%	11.019%	8.836%	5.1159
Algo3	57.562%	23.286%	8.972%	6.559%	21.19954
Algo4	58.499%	25.989%	10.216%	7.524%	24.79747
Algo5	55.244%	30.546%	19.771%	15.518%	11.66592
Algo6	57.657%	27.007%	12.809%	9.804%	25.8069

Table 3: Comparison Details of Click through Rate Grouping

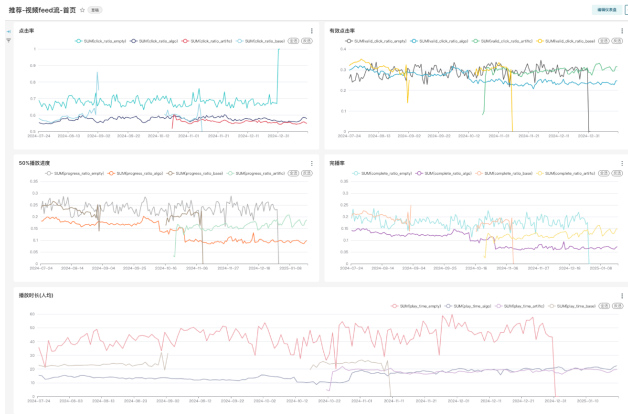


Figure 3: Summary and Comparison of Click through Rates

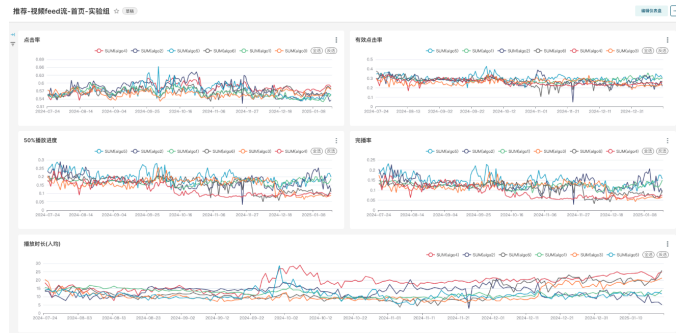


Figure 4: Comparison of click through rate groups

GROUPS	Order Counts	Total Counts	CVR
Algorithm Group	8528	31611	26.98%
Artificial Group	39010	161963	24.09%

Table 4: Summary and Comparison Details of Conversion Rates

GROUPS	Order Counts	Total Counts	CVR
Algo1	1214	4071	29.82%
Algo2	1040	3045	34.15%
Algo3	265	1026	25.83%
Algo4	7495	27427	27.33%
Algo5	208	779	26.7%
Algo6	400	1497	26.72%

Table 5: Comparison of Conversion Rate Grouping Details

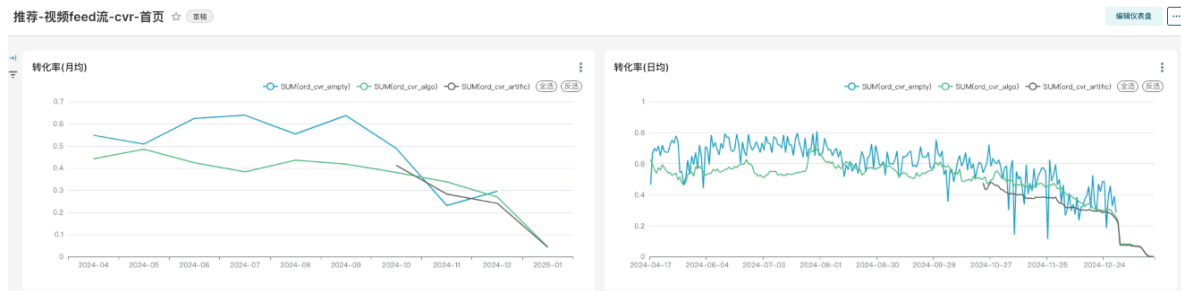


Figure 5: Summary and Comparison of Conversion Rates

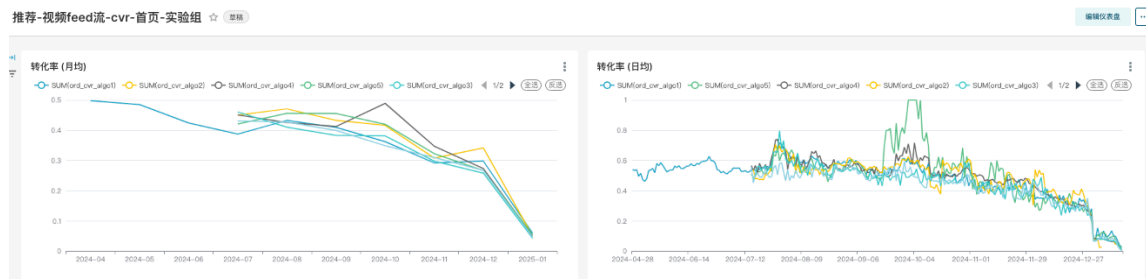


Figure 6: Comparison of Conversion Rate Grouping

From the experimental results, it can be seen that compared with the baseline models YouTube DNN, DSSM, DIN, ESMM, MMoE, and the comparison table of

experimental data, the indicators of the new algorithm are better than those of the single baseline model. The business indicators of the algorithm group have reached the

replacement baseline traffic standard, and the algorithm system has been expanded through engineering to support the full traffic standard.

6 LARGE SCALE DISTRIBUTED MACHINE LEARNING

6.1 Framework

(1) Data parallelism: training sample data slices, the size of a single slice depends on the machine memory size and the number of feature dimensions, with the maximum limit being that the training matrix does not exceed the total machine memory; In the process of data parallelism, after data is divided into blocks, each block of data is separately trained as a local model, and ultimately multiple small models are merged into a global model. During this period, the global model will be fed back to update each local model. The possible problem that may arise is that the small model itself is relatively large and cannot be loaded into the single machine memory. Therefore, we have introduced a parallel model approach.

(2) Model Parallelism: The standalone version of Python can utilize common parallel libraries such as the dask and Ray parallel libraries to transform pandas.DataFrame, directly achieving parallelism by parallelizing the basic DataFrame matrix units at the underlying level. Another approach is to directly base on the distributed training framework Spark, where some common machine learning algorithms (such as XGB) can be implemented using library functions; The other part of custom training algorithms requires manual partitioning, and the model is distributed to different worker nodes after partitioning. Each worker works independently and has no dependency on time scheduling, allowing for parallel execution and maximizing the use of idle shard space.

(3) Pipeline parallelism: The scheduling task pipeline is divided into blocks and parallelized, allocating different layers of the model to different devices, using micro batch processing to improve device utilization. By modifying the configuration of the main control process for scheduling tasks, such as dag or job, and modifying the execution mode on the scheduler, a certain number of N parallel blocks can be implemented, where each block executes sharding tasks independently without interfering with each other. This method allows for parallel execution of calculations between multiple machines in a pipeline manner. In the case of large models or limited network bandwidth, the training speed of distributed models will slow down because a lot of time is spent on network communication between nodes. The purpose of the pipeline parallel approach is to reduce

communication volume, allowing model training and inter node communication to occur simultaneously, thereby achieving high CPU/GPU utilization.

6.2 Development Stage

Stage 1: Single cluster distributed, trained within a single cluster, belonging to the same local area network, nodes belonging to the same network segment, and communication between each other does not require encrypted computation to protect privacy data. This situation applies to most of the aforementioned technologies.

Stage 2: Distributed across multiple local area networks/remote clusters, the model is trained on nodes in different local area networks that do not belong to the same network segment. Communication issues need to be addressed, and local data privacy protection mechanisms need to be introduced.

Stage 3: Large scale distributed model training, remote cluster, requiring the introduction of federated learning^[8] and privacy computing, cross domain joint modeling and training, while involving computing hardware: CPU/GPU/TPU heterogeneous computing, multi machine multi card, distributed training.

6.3 Industry research

The industry's focus on large-scale distributed machine learning and deep learning is currently mainly in the field of language model training, which mainly addresses the demand for computing power and memory in large models. On the one hand, it is solved through distributed clustering, and on the other hand, it is improved through enhancing hardware capabilities. The required computing power for the model is approximately six times the number of large model parameters multiplied by the number of tokens used for training. And hardware also has certain limitations in the industrial manufacturing field, which makes it difficult to break through the capacity of video memory. Usually, hardware utilization is positively correlated with the size of model parameters, and more training calculations can promote an increase in resource utilization. The concept of Spark data graph is proposed to allocate resources reasonably to distributed devices, and hardware companies such as NVIDIA also have large-scale pipeline parallel training related work. The core technology of Microsoft's DeepSpeed framework is aimed at solving the memory consumption of large-scale distributed training, mainly optimizing the allocation of fragmented data resources between model computations, and optimizing the forward and backward computing processes.

7 CONCLUSION

The experimental results of engineering performance indicators and algorithm models show that our new algorithm is superior to the mainstream baseline models in the current industry, and can adapt to data from different ends simultaneously, including traditional toC user end and toB merchant end. And due to the widespread adoption of distributed training after system upgrades, common evaluation metrics such as model training efficiency and performance, system throughput, etc. are superior to ordinary standalone versions.

8 REFERENCES

- [1] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 191-198.
- [2] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data.
- [3] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1059–1068. ACM.
- [4] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 5941–5948.
- [5] Weinan Xu, Hengxu He, Minshi Tan, Yunming Li, Jun Lang, Dongbai Guo. 2020. Deep interest with hierarchical attention network for click-through rate prediction.
- [6] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, Kun Gai. 2018. Entire space multi-task model: an effective approach for estimating post-click conversion rate.
- [7] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts.
- [8] Yang Liu, Tianjian Chen, Qiang Yang. 2018. Secure federated transfer learning.
- [9] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Qiang Yang. 2019. SecureBoost: a lossless federated learning framework.