Intel China

2020 AI Implementation Guide for Financial Industry

CONTENTS

Note: The sections marked with * are 2020 updates

Release Notes:

In addition to detailed revisions to the 2019 version, the Intel China 2020 AI Implementation Guide for Financial Industry has been supplemented with the following content:

- Intel® Optane™ Persistent Memory combines innovative algorithms and databases to provide a highly available, low TCO AI implementation solution for financial data.
- Intel® Software Guard Extensions brings hardware-based security environment to the financial industry, enabling users to explore the use of multi-source data in AI applications with a federated learning approach.
- A variety of Intel's advanced hardware and software products provide complete support for deep learning-based knowledge graphs, enabling financial industry users with the ability to obtain deep and high-value insight.
- Intel's end-to-end unified big data and AI platform facilitates building big data platforms and AI capabilities in the financial industry.

Editor-in-Chief: Yu Wei and Chen Zhiwen

Authors (listed by first name):

Hu Ying, Le Pengfei, Li Zhiqiang, Wu Guoan, Xia Lei, Yuan Chao, Zang Zhan, Zhao Yuping, Parviz Peiravi, Cao Jin, Gong Yimin, Lu Yi, Lu Liming, Shen Feilian, Sun Yu, Wang Dongfang, Wei Jian, Yi Hongwei, Zhu Lejun

In addition, we would like to thank our partners and many Intel colleagues for their help in preparing this guide.

V.2020.002

Trends

Artificial Intelligence Empowering Financial Industry Innovations

After years of evolution, Artificial Intelligence (AI) is stepping into a new era. More and more companies are choosing this technology which brings a significant impact to our economy, society and lifestyle, to start their brand-new digital transformation. Particularly when we look at the financial industry, it's easy to see that over the past decade or so, the industry's bellwethers have been investing more and more money in big data, robotics and cloud computing services, and these efforts have also been highly welcomed by investors. A report from Sina Finance said that[1]:

> Back in 2000, the U.S. cash equities trading desk at Goldman Sachs's New York headquarters employed 600 traders, buying and selling stock on the orders of the investment bank's large clients. Today there are just three equity traders left.
>
> According to statistics from Business Insider, Goldman Sachs currently employs 33,000 full-time employees, of which more than 9,000 are programmers and engineers. In recent years, the CEO of Goldman Sachs has repeatedly stated that Goldman Sachs was really a tech firm, not a bank.
>
> ...
>
> Another Wall Street giant JPMorgan Chase has taken a similar approach. JPMorgan Chase set up a technology center long time ago, employing about 40,000 technicians to research big data, robots and cloud infrastructure, with a technology budget of 9.6 billion US dollars, accounting for 9% of its total revenue. Last year, the company also announced the use of the world's first robot to conduct their algorithm trading in global stock market. Before that, LOXM, an AI project developed by JPMorgan Chase in Europe, had already benefited from similar trading.

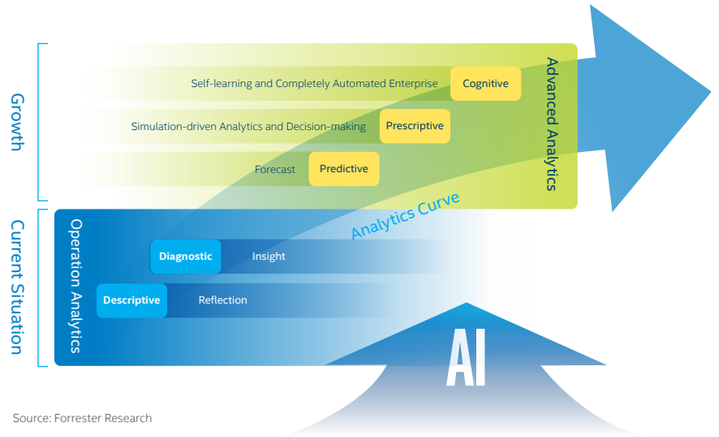
So, why has AI become a new hot topic in the financial industry? The reason is that AI and the financial industry inherently have several common characteristics. First of all, both AI and finance are built on massive data, which provides abundant data resource for AI model training and predictive inference. Secondly, AI can drastically reduce service costs spent by traditional financial enterprises on customer relationship management. In addition, AI can also enable financial enterprises to take more targeted actions and further improve the quality of their business. And more importantly, the combination of many innovative financial services with AI can leave more space for the development of the industry and more possibilities for business innovation.

Taking banks as an example, by integrating with AI and relying on their huge business data, they have the capability to innovate their data analysis and prediction models in an intelligent way, which will allow them to gain new insights, build more sensitive and efficient business models and avoid risks such as overdue loan and illegal fraud, thus taking the lead in future competition.

However, the evolution from data analytics to AI cannot be achieved overnight. As shown in the figure below, the evolution of enterprise data analytics technology normally goes through five phases where the scale and maturity grow incrementally. These five phases are descriptive, diagnostic, predictive, prescriptive and cognitive. Mature AI capability will empower all data analytics phases above, driving data analytics to a higher level of maturity and a larger scale.

In recent years, AI has witnessed rapid breakthroughs in computing power, algorithms, and data capabilities, including:

- **Improvement in computing power:** driven by Moore's law, chip process and architecture innovation are enabling continuous breakthroughs in computing power to meet the computing demand of compute-intensive algorithms used



Source: Forrester Research

Figure 1-1-1 Evolution of Enterprise Data Analytics Technology

[1] For detailed report, please refer to: http://finance.sina.com.cn/world/2018-05-22/doc-ihawmaua4464623.shtml

by machine intelligence. For example, the concept of deep learning through artificial neural networks was invented at least 20 years ago, but until recent years when the computing technology improved to provide high-precision, high-speed computing capabilities required by AI, these compute-intensive algorithms can be applied in practice.

- **Emergence of innovations:** it is not enough for AI to be driven by computing power and data alone. There is no doubt that the key force that drives AI to solve real-world problems is innovation, which brings AI from research to real-world use cases. It has proved that every innovation of AI algorithm creates more possibilities for its application and attracts more innovators to join the rush into application development. For example, the neural network innovation in the 1990s has driven the re-elaboration and research of AI. In 2009 and 2012, breakthroughs were made in the fields of speech recognition and image recognition respectively, which have also become the catalysts for the current AI innovations.

Meanwhile, massive data is also a driver for the rapid development of AI in the financial industry. On the one hand, the wide adoption of Internet of Things (IoT) and the rapid growth of interconnected smart devices have generated an increasing amount of structured and unstructured data. Studies have shown that by 2020, the amount of data generated by smart, interconnected devices in the IoT could reach 40 ZB (1 ZB = 1 billion TB)[2]. Such a huge amount of data undoubtedly lays the cornerstone for training AI algorithms and uncovering new insights in the financial industry. On the other hand, more methodologies and hardware products have emerged for the processing and use of different types of financial data. For example, in response to the high-dimensional nature of financial data, enterprises can choose more targeted high-dimensional algorithms, high-performance real-time feature databases, and storage devices with better caching performance and higher capacity to build AI applications, reduce TCO and improve processing performance.

It is worth mentioning that although everyone agrees that massive amount of high quality data can dramatically improve core competencies of AI. However, in the financial industry, companies often build a series of stringent safeguards to avoid the risk of data breach, which also leads to the creation of data silos. To make the interaction, transmission and aggregation of multi-sourced data more secure, new collaborative learning approaches including federated learning are adopted to build more secure and trusted multi-sourced data co-training solutions that provide richer data sources for AI applications to improve their accuracy while ensuring data security.

The federated learning solution built on the Trusted Execution Environment (TEE) such as Intel® Software Guard Extensions (Intel® SGX), is now being used in insurance pricing, credit risk management, sales forecast and other financial areas. Using insurance pricing as an example, the solution can provide secure and compliant access to multi-source data while protecting the private data of each partner's users from leaving their local domains. Therefore, on the one hand, it enables insurance companies to develop personalized pricing strategies with multi-source and multi-dimensional user behavior data; on the other hand, it effectively identifies insurance frauds with the help of security-related multi-source big data. Some research show that insurance data models built with federated learning can create a more detailed risk profile system, thereby significantly improving the accuracy of insurance pricing.

Thanks to these improvements in computing power, algorithms and data processing technologies, AI is becoming an important tool for financial enterprises to conduct high-quality data analysis and business forecasting. Now, with the continuous development of AI technology, there are relatively mature front, middle and back end solutions for the financial industry.

At the front end, perceptual technologies (such as computer vision, speech recognition and natural language processing) continue improving, and their typical applications include customer service chat robots and automatic identification.

The customer service chat robot can follow the standard path of interaction with customers, observe its dialogue with customer and understand customer's intention with the aid of machine learning algorithm, send customer's questions to a service representative when it gets stuck, and learn from their replies, thus continuously improving customer service quality and reducing service cost. On the other hand, automatic identification is able to analyze a user's voice, eyes and facial features through voice recognition, facial recognition and other technologies to verify the user's identity. This type of AI authentication is more efficient than the traditional security questions or password authentication, and does not require users to remember their passwords, thus greatly improving customer experience.

At the middle end, AI can improve the efficiency of information-based analysis and decision-making and help users seize business opportunities more quickly. Existing Business Intelligence (BI) and traditional data analytics still rely on approaches such as trend analysis, cause mining, data mining and prediction. However, the introduction of AI not only extends the scope of analysis, but also improves the depth of analysis. AI can improve the relevance and specificity of its suggestions through continuous learning and improvement, perform "personalized analysis", and provide intelligent analysis and decision-making for risk management, marketing, and services.

For example, based on credit scores from social network, AI can optimize existing scores or score users without credit records. In addition, AI can generate analysis reports through Natural Language Processing (NLP) techniques as well as analyze and evaluate financial data. At the same time, AI can also carry out dynamic fraud detection to discover and avoid risks from real-time complex transactions, and provide personalized financial health advice through customer behavior research. Moreover, by using AI, financial companies can also achieve personalized marketing and provide unique personalized services based on customer and product DNA.

[2] For detailed report, please refer to: https://cloud.cent.com/info/5a0335164080e583fd6a2d4a735e7def.html

By introducing new collaborative learning methods, such as federated learning, AI can horizontally aggregate more training data to improve model accuracy, thus leading to enterprise-level data collaboration across geographies and sectors, and allowing all participants to benefit from the increasing AI capabilities in a "win-win" manner.

At the back end, there are a lot of highly repetitive tasks such as industry compliance and IT, accounting and other support functions. One of the important applications of AI is to gradually take over these repetitive tasks. Therefore, AI has a huge potential to empower the back-end support process.

In short, regarding AI capabilities at the front, middle and back ends, the AI application scenarios in the financial industry are currently focused on four aspects, i.e. risk & compliance, customer experience, marketing decision-making and intelligent operation.

As shown in the figure below, risk and compliance applications mainly involve how to deal with fraud risk, credit risk, macro risk, anti-money laundering and how to analyze compliance policy documents, and typically use AI technologies such as machine learning, deep learning-based face recognition and voice recognition and knowledge map. Customer experience applications mainly include intelligent investment consulting, intelligent claim settlement, intelligent customer service, identification, and agricultural and animal recognition, and typically use AI technologies such as deep learning-based face recognition and voice recognition, behavior pattern recognition, NLP and robot technology. Marketing decision-making mainly involves customer profiling, targeted marketing, smart recommender, credit scoring, capital flow monitoring, and quantitative analysis, and typically uses AI technologies such as deep learning and machine learning. Intelligent operation applications mainly include intelligent channel management, receipt recognition, intelligent operation and maintenance, and typically use AI technologies such as deep learning-based image recognition and knowledge map.

Figure 1-1-2 AI Application Scenarios in Financial Industry

In the next chapter, "Implementation", we will elaborate several actual AI implementations deployed by China UnionPay, China Life Insurance Shanghai Data Center, MasterCard, 4Paradigm, China Ping An, INTSIG and other partners in financial anti-fraud, risk prediction, customer marketing, smart recommender, intelligent insurance settlement and other scenarios, and discuss the technical details involved, especially the application and optimization of Intel-based technologies and products in these real-world scenarios, and the best practice recommendations for hardware and software configuration.

Implementation

Big Data + AI-based Efficient Real-time Financial Anti-fraud Solution

Intel cooperates with financial companies to learn user behavior features using RNN model

Evolution of Anti-fraud Model

During our cooperation with financial customers, we found that the existing design of anti-fraud model in the financial industry often has the following problems:

- The algorithm for learning user behavior lacks sufficient applications and uses.
- Traditional deep learning solutions require huge amount of data, but financial companies are unable to provide historical transaction data corresponding to each user behavior pattern according to the algorithm.
- The data imbalance ratio is very high, i.e. the vast majority of training data come from normal trading behavior and the normal/abnormal data ratio is about 100,000 ~ 1 million to 1.

Traditionally, financial companies and institutions often adopt a rule-based approach to build their risk control and anti-fraud model, which is characterized by constantly creating and updating a repository of rules based on user behavior features. When a transaction occurs, the system will monitor the potential risks of the transaction by using the rule engine.

For example, if a business person often goes abroad, his frequent overseas transactions will be identified as normal by the transaction rule repository. However, if an elderly person rarely uses his credit card outside his place of residency, his normal transaction behavior may be characterized with multiple small and local transactions, so when he makes large transactions, this abnormal transaction characteristic will be captured and alerted by the rule engine. If his account repeatedly has abnormal large transaction records in different places, the account may be monitored by the risk control system and carried out with follow-up verifications.

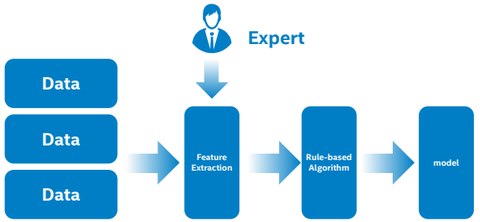


Figure 2-1-1 Traditional Anti-fraud Model

Although the rule-based risk control system is effective, but as a retrospective system, the rule repository needs to continuously review and summarize existing business cases. This means that users will spend a lot of resources at set intervals to summarize the business cases and update the rules. However, as the business scenarios increase, the complexity of transaction rules will also grow, which continues to increase the resource consumption and monitoring delay in the risk control system. Therefore, financial institutions attempt to use AI to build a more efficient financial anti-fraud model by relying on machine learning, deep learning and other technologies.

Compared with the rule-based approach, the AI anti-fraud solution has higher reliability and accuracy, especially its ability to learn rules automatically. In layman's terms, the rule-based approach will tell the system in advance that method A and method B are wrong and trigger an alarm if a method is wrong. On the other hand, machine learning and deep learning approaches will use huge historical data as samples for learning and perform a lot of training by using a large number of computing units, thus obtaining an evaluation model. When a new transaction enters into this model, the system can determine the legitimacy of the transaction itself.

Figure 2-1-2 Typical Algorithms Used by Current Anti-Fraud Models

As shown in Figure 2-1-2, some excellent machine learning classification algorithms such as Logistics Regression (LR), Random Forest (RF) and Gradient Boosting Decision Tree (GBDT), have been widely used in anti-fraud models.

RNN-based Deep Learning Approach

Deep learning is a common solution for AI anti-fraud, while Recurrent Neural Networks (RNN) is one of the mostly used deep learning models in financial anti-fraud. A typical RNN structure is shown in Figure 2-1-3 below. RNN will give an output for each moment's input combined with the state of the current model. As you can see from the figure, in addition to X from the input layer, the input of RNN's main structure A also has a cycle to provide the state at the current moment, and at the same time, the state of A will be transferred from the current step to the next step.

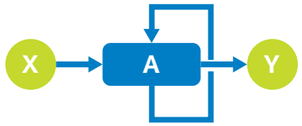


Figure 2-1-3 Typical RNN Structure

In practice, RNN also has two important variants, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTM can avoid the long-term dependence problem in the classical RNN network structure by designing three special "gate" structures, thus greatly improving the memory duration. The "gate" in LSTM is the combination of Sigmod neural network and bit multiplication. The fully connected neural network layer with Sigmod as the activation function will return a value between 0 and 1, describing the amount of input information that can pass through the structure. When the output of Sigmod is 1, all information can pass through. On the contrary, when the output of Sigmod is 0, no information can pass. GRU is an improved version of LSTM, which combines the three "gate" structures of LSTM into two, and is different from LSTM in calculating the current new information.

RNN-based deep learning approach has always performed well in sequence-based analysis. With the help of sequence tagging technology, RNN deep learning approach can be used to analyze the real-time behavior status of different accounts.

However, the single deep learning technology is not ideal for modeling fraud detection against a large number of transaction data. The reason for this is that although RNN and other neural networks can learn the feature correlation between transaction sequences, they do not have sufficient ability to learn the features within a single transaction and cannot reach the expected goal.

For example, when detecting fraud, the combination of features such as "large over-the-counter transaction occurring at midnight" may indicate a very suspicious transaction behavior, while "over-the-counter", "large transaction" and "midnight" are all individual common features. This kind of combination of features may not be trained properly by using a single deep learning technology.

Model Implementation and Optimization

RNN deep learning approach can be implemented using deep learning frameworks such as Keras and TensorFlow. A variety of frameworks such as TensorFlow, Theano and CNTK, can be used as the back end of Keras. When developing the following algorithm model, TensorFlow is selected as the back end of Keras. You can set it by modifying relevant parts in **$HOME/.keras/keras.json**:

```
"backend": "tensorflow"
```

■ Installation of Intel® Optimization for TensorFlow

TensorFlow is one of the mainstream frameworks in the field of deep learning. It allows deep learning developers to use computing resources and build deep learning models more efficiently. In order to make full use of Intel® architecture and achieve higher performance, the TensorFlow framework has been fully optimized by using the Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN).

Anaconda is an open source Python release that helps users build TensorFlow using Intel® MKL-DNN (supported from TensorFlow v1.9) to provide higher performance for Intel® processors. If users currently use Conda package manager to manage their environment and packages, they only need to install the TensorFlow package from **Anaconda.org** into the virtual environment.

The command to install TensorFlow in Anaconda is as follows:

```
conda install tensorflow
```

If the user's Anaconda pipe is not the default pipe with the highest priority, the following command can also be used to obtain Intel® Optimization for TensorFlow.

```
conda install -c anaconda tensorflow
```

In addition to the above installation methods, Intel® Optimization for TensorFlow can also be distributed as wheel and docker images or as Conda package.

Moreover, users can use the following command to install Intel® Optimization for TensorFlow in the existing Python environment by using PIP:

```
pip install intel-tensorflow
```

Or users can install wheel in the existing Python environment and are recommended to use Intel® Distribution for Python. The commands to install version 1.13.1 in Python 2.7 and Python 3.6 are:

```
# Python 2.7
pip install https://storage.googleapis.com/intel-optimized-tensorflow/tensorflow-1.13.1-cp27-cp27mu-linux_x86_64.whl
# Python 3.6
pip install https://storage.googleapis.com/intel-optimized-tensorflow/tensorflow-1.13.1-cp36-cp36m-linux_x86_64.whl
```

■ Intel® Optimization for TensorFlow

Intel® Optimization for TensorFlow provides a series of optimizations that can effectively improve the efficiency of models. These optimizations include adjusting the number of processor cores, and introducing the Non-Uniform Memory Access Architecture (NUMA) technology and Intel® MKL-DNN. The optimization steps are as follows:

■ Setup Environmental Variables

First of all, uses need to set environment variables. The commands include: clear the system cache, set the processor to the performance priority mode, i.e. running at the highest frequency, and turn on turbo boost for the processor. The setup commands are as follows:

```
echo 1 > /proc/sys/vm/compact_memory
echo 3 > /proc/sys/vm/drop_caches
echo 100 > /sys/devices/system/cpu/intel_pstate/min_perf_pct
echo 0 > /sys/devices/system/cpu/intel_pstate/no_turbo
echo 0 > /proc/sys/kernel/numa_balancing
cpupower frequency-set -g performance
export KMP_BLOCKTIME=1
export KMP_AFFINITY=granularity=fine,verbose,compact,1,0
export OMP_NUM_THREADS=20
```

- Set **KMP_BLOCKTIME** to 1 to specify the time, in milliseconds, that a thread should wait, after completing the execution of the current task, before sleeping.
- Set **KMP_AFFINITY** to Compact, which means that in this mode, thread binding takes precedence according to the computing requirements of the computing core, i.e. binding the same core first, and then sequentially binding the next core on the same processor. This kind of binding is suitable for the computation with data exchange or common data between threads. Its advantage is that it can make full use of the characteristics of multi-level cache.
- Set **OMP_NUM_THREADS** to 20 to specify the number of parallel execution threads to a certain number of physical cores.

■ Add Thread Control to the Test Code

The code to add thread control is as follows:

```
config = tf.ConfigProto()
config.allow_soft_placement = True
config.intra_op_parallelism_threads = FLAGS.num_intra_threads
config.inter_op_parallelism_threads = FLAGS.num_inter_threads
```

As shown in the above code, during initialization of **tf.ConfigProto()**, we can also control the number of threads that each operator computes in parallel by setting the **intra_op_parallelism_threads** parameter and the **inter_op_parallelism_threads** parameter. The difference is:

- **intra_op_parallelism_threads** controls operator internal parallelism. When there is a single operator, and the internal can implement parallelism, such as matrix multiplication, and **reduce_sum**, you can set the **intra_op_parallelism_threads** parameter to implement parallelism (intra means internal).
- **inter_op_parallelism_threads** controls parallel computation among multiple operators. When there are multiple operators and they are independent of each other, there is no direct Path connection between operators. **TensorFlow** will try to compute them in parallel using a pool of threads, whose number is controlled by the **inter_op_parallelism_threads** parameter.

Normally, **intra_op_parallelism_threads** is set to the number of physical cores of a single processor, while **inter_op_parallelism_threads** is set to 1 or 2.

■ Utilize the Characteristics of NUMA to Control the Use of Processor Computing Resources

The servers used in data center are usually equipped with two or more processors, and most of data centers use NUMA technology to run many servers as if they were a single system. A processor can access its own local memory faster than non-local memory. In order to obtain better computing performance in such system, it needs to be controlled by some specific instructions. **numactl** is a technical mechanism for controlling processes and shared storage. It is a widely used method for controlling computing resources in Linux. The specific usage is as follows:

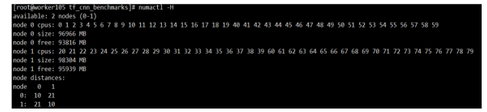


Figure 2-1-4 Utilize the Characteristics of NUMA to Control the Use of Processor Computing Resources

Execute the **numactl** command using Python command as follows:

```
numactl -C 0-19,40-59 -m 0 python3 test.py
```

The command indicates that when executing, **test.py** only uses the 0-19 and 40-59 cores in processor **#CPU0** as well as only uses the near-end memory corresponding to processor **#CPU0**.

■ Use Keras/TensorFlow to Implement Multilayer LSTM/GRU

The code for implementing multilayer LSTM/GRU in Keras is as follows:

```
classifier.add(GRU(number_of_cells=128,
    input_shape=(timesteps, data_dim),
    recurrent_dropout=0.3,
    activation='sigmoid',
    recurrent_activation='hard_sigmoid',
    return_sequences=True))
classifier.add(GRU(number_of_cells=64,
    recurrent_dropout=0.3,
    activation='sigmoid',
    recurrent_activation='hard_sigmoid'))
```

* For more technical details about Intel® Optimization for TensorFlow, please refer to relevant content in the Technologies section of this guide.

Sandwich-structured Fraud Detection Model

■ Model Introduction

The "GBDT->GRU->RF" three-layer structure is shown in Figure 2-1-5. This framework is designed to address the deficiencies of single deep learning technology (e.g. RNN) when learning features within a single transaction. With the help of Analytics Zoo tool provided by Intel, this framework uses GBDT model as its front end to optimize features. The optimized features and artificial features will be combined as input for the GRU network to enable learning of inter-sequence features and sequential features within a single transaction. This process can effectively filter the data, thereby providing truly useful data for the subsequent GRU model.

In the middle layer, the framework does not directly use the GRU network output to distinguish fraud detection, but rather as a step-in inter-sequence feature learning. The inter-sequence features learned will be combined with the original features in the transaction to form a final transaction feature vector.

On this basis, in order to further conduct combined learning with the sequential features, the framework will lastly add a top layer on the RF model as a final classifier to distinguish fraud.

■ Software Stack

The software stack of sandwich-structured multi-layer fraud detection model is shown in Figure 2-1-6. Within the software stack, the bottom layer is the hardware infrastructure built on Intel® Xeon® Gold Processor (6000 series). On top of it is RedHat Linux operating system (CentOS 7.4 Kernel 3.10.0-957.12.1.el7.x86_64), and virtual machines that are created by virtualization software. Intel® MKL-DNN or Intel® MKL is deployed on the virtual machines, and Intel® MKL-DNN optimized TensorFlow 1.10 and Intel® Distribution for Python are also installed. Fraud detection applications are deployed at the top layer.

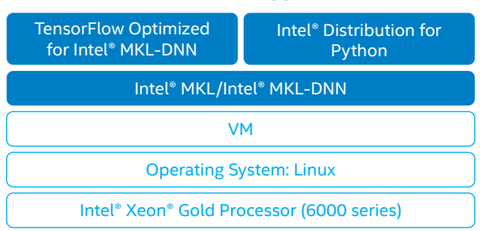


Figure 2-1-6 Software Stack of Sandwich-structured Multi-layer Fraud Detection Model

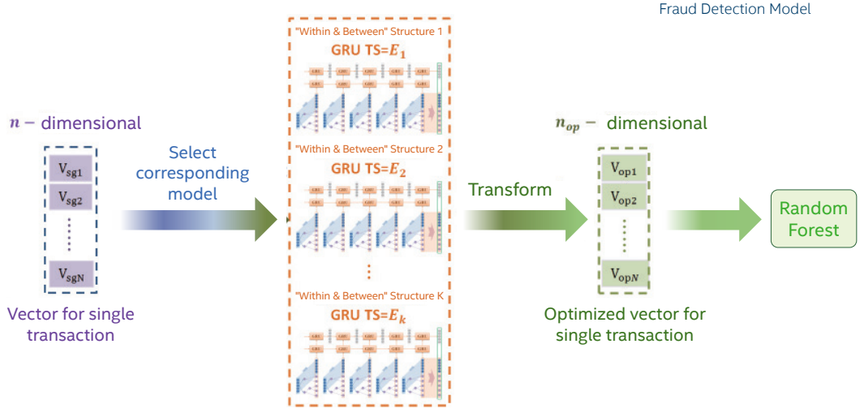


Figure 2-1-5 GBDT->GRU->RF Sandwich-structured Anti-fraud Model[3]

[3] For technical details about the sandwich-structured multi-layer fraud detection model, please refer to: Transaction Fraud Detection Using GRU-centered Sandwich-structured Model

■ Install Intel® Distribution for Python

The core package of Intel® Distribution for Python can be installed in Python 3/Python 2 environment by executing the following commands:

```
conda create -n idp intelpython3_core python=3
conda create -n idp intelpython2_core python=2
```

Or install the complete package of Intel® Distribution for Python:

```
conda create -n idp intelpython3_full python=3
conda create -n idp intelpython2_full python=2
```

Activate application environment:
Under Linux/MacOS:

```
source activate idp
```

Under Windows:

```
activate idp
```

Use the Conda command to install additional software packages, such as sympy.

```
conda install sympy
```

For details about the above procedure, please refer to:
https://software.intel.com/content/www/us/en/develop/articles/using-intel-distribution-for-python-with-anaconda.html

* For more technical details about Intel® Distribution for Python, please refer to relevant content in the Technologies section of this guide.

■ How to Solve Data Imbalance Ratio

The data imbalance ratio is a common problem in anti-fraud applications. In some scenarios, the ratio of fraud samples to normal samples is as high as 100,000-100,0000 to 1. During sampling and training, the following principles can be used to address data imbalance:
1. Due to the huge number of normal samples, normal sampling is sufficient to meet the training needs for normal samples.
2. During training, we can increase the weight of data of fraud samples.

```
class_weights= dict()
class_weights[0] = 1  #正常样本权重
class_weights[1] = 20  #欺诈样本权重
classifier.fit(X_train,
    y_train,
    batch_size=BS,
    epochs=runEpoch,
    class_weight=class_weights)
```

3. Different from data such as images and voice that are easy to re-label, fraud data is difficult to generate and label. Therefore, we still use random disorder and train fraud samples many times, but only need to use normal samples once to further solve the data imbalance problem.

■ Methods to Improve Algorithm Accuracy

■ Order of Combination of Different Techniques are Correlated with Accuracy:

When combining intra-feature extraction algorithm with inter-feature extraction algorithm, different order of combination will lead to different level of accuracy. Our tests show that adding an inter-feature extraction technique between two intra-feature extraction techniques can achieve the highest level of accuracy.

■ Sandwich Structure Uses Bypass to Improve Feature Reuse:

Like the Densenet algorithm, the sandwich structure also constructs connection relationship between different methods. This bypass method can enhance the overall training effect by improving feature reuse.

Recommended Software Configuration

When modeling training data flow and verifying multi-layer anti-fraud model construction, you can refer to the following software configuration built on Intel®-based platform.

Name	Specification
Operating System	Centos7.4
Linux Kernel	3.10.0-957.12.1.el7.x86_64
Workloads	GRU
Compiler	GCC5.4
Library	Latest version of Intel® MKL-DNN
Framework	Intel® Optimization for TensorFlow Distribution
Hadoop Distribution	Cloudera CDH-5.9.0. or higher version
Spark version	Apache Spark-2.1.0. or higher version
Other Software Configurations	Analytics Zoo Intel® Distribution for Python

Use Case: China UnionPay

Background

With rapid business expansions in the financial industry, the risk index has also risen sharply. Especially for bank cards and credit cards, the fraud loss rate is increasing year by year while the amount of fraud loss is also rising. Therefore, anti-fraud is becoming an important part of risk control in the financial industry.

In terms of risk forms, traditional risks and new risks are also intertwined. In addition to the incessant emergence of traditional financial fraud methods, such as credit fraud, fraudulent use, malicious cash-out and insurance fraud, the problems of personal information disclosure, phishing websites and fraudulent black market in the Internet era have also brought more highly-frequent and accurate financial fraud crimes.

In order to address this problem, financial institutions have developed many meticulous anti-fraud solutions. With the development of information technology, especially the continuous breakthrough of AI technology, more and more AI capabilities are being combined with financial risk control systems to build more effective anti-fraud models.

As a financial institution that provides professional banking and payment services and has the largest market share in card issuance and transaction volume in the world, China UnionPay (hereinafter referred to as "UnionPay") is sparing no effort to introduce AI capabilities and build an efficient financial anti-fraud model. In this case, China UnionPay and Intel jointly researched deep learning-based anti-fraud technologies.

By integrating experience gained from anti-fraud models such as rule-based and machine learning, China UnionPay has built an efficient fraud detection solution by using a sandwich-structured multilayer model, with the help of all-round optimizations provided by Intel architecture. At present, the solution has been tested in scenarios such as fake card/ cash-out fraud detection and has achieved good results.

Solution

China UnionPay built an efficient anti-fraud model by using a GBDT→GRU→RF sandwich structure. First of all, UnionPay performs data process modeling by using Analytics Zoo and Spark pipeline. Normally, AI training models need to analyze each transaction and behavior of users, in other words learn the consumption behavior pattern of each cardholder through algorithms, thereby analyzing whether there are abnormalities and start interception actions when abnormal transaction behaviors are found, but this requires the system to work with massive transaction data. At the same time, the training model needs to learn the user's historical transaction behavior, while the system is required to provide at least hundreds of abnormal transaction data for each user to be learned by the model.

For this reason, UnionPay has built a mass data storage platform based on Hadoop and introduced tools such as Analytics Zoo to perform process modeling against the training data. In order to solve the problem of insufficient data for learning historical transactions, the platform can derive hundreds of feature factors from a small number of original fields by using the modeling process, which can be generalized into 6 dimensions such as current transaction/ last transaction, long/short term statistics and reliable feature variables, and enable the model to learn better through these feature engineering. Then, based on a GBDT->GRU->RF three-layer model, UnionPay built its fraud detection model on a training cluster composed of hundreds of nodes, and achieved good results.

Through multi-dimensional evaluation, the new multi-layer anti-fraud model has achieved the expected results in terms of both recall rate and accuracy rate. Compared with traditional machine learning approach or standalone RNN approach, F1 Score (a weighted average of accuracy rate and recall rate used to measure the performance of the detection model) has improved dramatically, exceeding the critical point of business deployment.

As shown in Figure 2-1-7, in the left, compared with other machine learning + deep learning models or multilayer models, the GBDT→GRU→RF sandwich-structured anti-fraud model has the best Precision-Recall (PR) curve (the uppermost curve represents test values of GBDT→GRU→RF sandwich-structured anti-fraud model). In the right figure, it can be seen that as the data imbalance ratio increases, the decline of F1 Score of the GBDT→GRU→RF sandwich-structured anti-fraud model is the slowest.

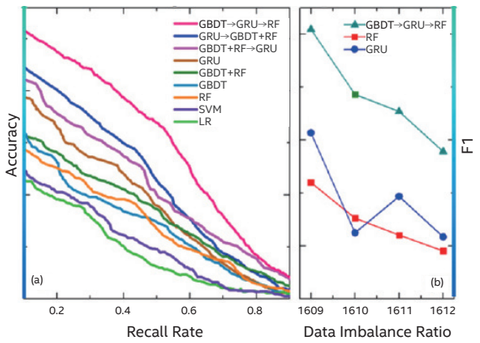


Figure 2-1-7 Evaluation Result of GBDT→GRU→RF Sandwich-structured Anti-fraud Model

After completing the process modeling and the multi-layer fraud detection model building, UnionPay packaged and integrated them together, and provided an intelligent end-to-end analysis solution as API interface to provide better services for its business departments. As shown in Figure 2-1-8, after providing input parameters through API interface or other methods, the user can obtain the result index analyzed and computed by the intelligent model.

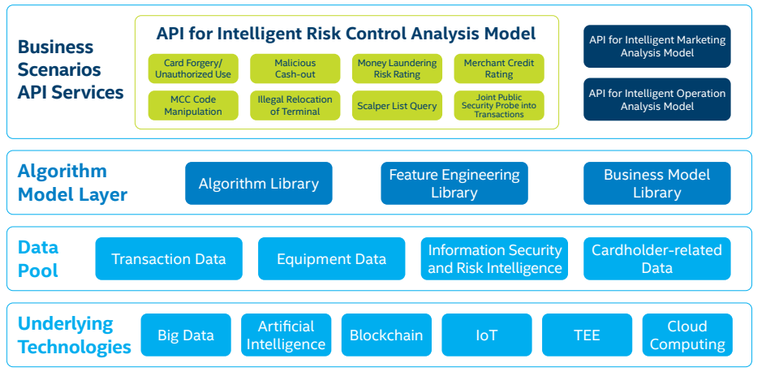


Figure 2-1-8 Architecture of Intelligent Analysis Service Platform Built by Electronic Payment Research Institute of China UnionPay

Taking the sandwich-structured fraud detection model as an example, it can provide underlying model support for fraud detection scenarios such as fake cards and cash-outs, while business departments only need to invoke the upper layer API according to data specification rather than study these complex models in depth.

The training cluster adopted by UnionPay is built on Intel® Xeon® processors. This platform not only boasts high-performance cores and caches, but also can improve the performance of the framework with a number of hardware enhancements. In addition to basic computing power, the platform also provides the following capabilities to facilitate the building of UnionPay AI anti-fraud model:

- Support for highly irregular computation, such as tree construction, entropy computation, tree traversal and reduction.
- Support for conventional computation, such as GRU, nonlinear activation and batch normalization.

At the same time, Intel® Advanced Vector Extensions 512 (Intel® AVX-512) technology integrated by Intel® Xeon® Processor/Intel® Xeon® Scalable Processor provides exceptional parallel computing capability for the sandwich-structured anti-fraud model of UnionPay.

Conclusion

To address the insufficiency of machine learning-based approach in learning features of serialized transactions and the limitation of standalone deep learning model on learning features within a single transaction, UnionPay and Intel jointly proposed a multi-layer machine learning+deep learning model to greatly improve the performance of the anti-fraud model through technological innovation.

During our cooperation with UnionPay, Intel not only provided high-performance processors as computational engine for this new anti-fraud model, but also offered diversified, scalable and all-round technical support by providing targeted optimizations and tools for the technologies used in each layer of the sandwich-structured fraud detection model, thereby further improving the efficiency of the entire anti-fraud model.

The hardware platform built on Intel® Xeon® Processor/Intel® Xeon® Scalable Processor as well as a number of optimizations from Intel provides strong computing power for the successful building and application of China UnionPay anti-fraud model. In the future, users can also choose newer hardware products such as the 2nd Generation Intel® Xeon® Scalable Processor with better performance and more optimizations for AI to build solutions with better performance.

AI-based Efficient Credit Risk Forecasting Solution

Intel collaborates with financial companies to predict credit risk using AI model

Credit Risk Challenge

Credit is one of the most important asset businesses of financial institutions. With the continuous expansion of the credit business scale of various commercial banks, the increasing non-performing loans not only gradually erode the bank's profits, but also occupy valuable credit lines, affect the bank's lending capacity, and make high-quality projects unable to obtain credit support.

What is more serious is that when non-performing loans exceed a certain limit, it will greatly affect the business operation and bring risks to the bank. In addition, the proliferation of non-performing loans will also incur social moral hazard. If the non-performing loans are disposed aggressively, it may cause chain bankruptcy of enterprises, increasing the probability of financial risk and social crisis.

Data from China Banking and Insurance Regulatory Commission (hereinafter referred to as "CBIRC") shows that by the end of the fourth quarter of 2018, the balance of non-performing loans of commercial banks in China was 2.03 trillion yuan, with a non-performing loan ratio of 1.83%[4]. Therefore, the implementation of efficient pre-loan and post-loan risk control for credit business has become an important part of the bank's risk control system.

At present, the credit risk forecasting of commercial banks includes two key scenarios. One is pre-loan risk assessment which focuses on timeliness and interpretability of the forecasting results, the other is the post-loan risk forecasting which focuses on the accuracy and interpretability of the forecasting results.

For pre-loan risk forecasting, commercial banks mainly assess the risk level of loan issuance by conducting multi-faceted manual research on the characteristics of the industry to which the enterprise belongs, as well as the actual operation, assets and liabilities and credit status of the enterprise. However, such approach is usually inefficient, requires a lot of efforts and time, and are accompanied by obvious subjective judgment.

For post-loan risk forecasting, commercial banks usually carry out manual on-site or off-site inspections periodically or irregularly according to the industry to which the enterprise belongs and operation characteristics of the enterprise, communicate with the enterprise to analyze its financial information and operating conditions, and monitor the flow of loans, thereby discovering the factors that may cause default risk and credit risk and avoiding loan defaults.

Due to limitation on manpower and cost, this manual approach can only be carried out once a month or once a quarter, and problems found can only be escalated level by level, waiting for the risk management department to take action after making their decisions. Therefore, it faces the following key problems:

- Huge labor input and long forecasting time. Every month, the bank staff is required to forecast the risk of overdue loans that will expire in the current month as well as in the following three months to six months, escalate them level by level and wait for decisions. The whole process will take nearly one month, wasting valuable time that could be spent on risk control in loan management.
- The quality of manual forecasting is spotty and uncertain. Some of the staff are very experienced and have capability to make accurate forecasting. Some are inexperienced and unable to eliminate risks in their infancy.
- Many factors affect the forecasting. In the process of manual forecasting, the timeliness and accuracy of risk forecasting will be affected by the variability of market environment, the periodicity of economic activities and the asymmetry of information between banks and enterprises.

Meanwhile, apart from the difficulties to ensure accuracy, the manual forecasting also lacks a sophisticated knowledge system to gradually improve the forecasting efficiency and accuracy through the accumulation of experience, which makes it unable to build a closed loop with positive feedback.

With the continuous expansion of credit business, the traditional manual risk forecasting approach adopted by commercial banks are also facing more and more challenges.

The above problems, as well as the monthly overdue loan monitoring and supervision by CBIRC, undoubtedly bring tremendous cost and management pressure to banks. Therefore, banks intend to utilize their abundant business data and build a more effective credit risk forecasting system using AI. However, in order to build a complete AI framework for credit risk forecasting and to develop a overdue loan forecasting solution with high accuracy, low delay and interpretability, it is necessary to analyze and predict business data and environmental data.

Business data is enterprises' financial asset status, future cash flow, and use of fund recorded by financial institutions. At present, the industry usually adopts machine learning or deep learning technologies to build prediction models. Environmental data can be studied and predicted by using Natural Language Processing (NLP).

During the cooperation between Intel and financial companies, we have jointly built a hybrid model based on LSTM and traditional machine learning to meet users' needs in terms of accuracy and interpretability. At the same time, we have also tried to build an NLP model for environmental data.

Architecture Design of Credit Risk Forecasting Model

■ Machine Learning-based Approach

The tree-based machine learning technologies are widely used in credit risk forecasting models, and their forecasting results are usually well interpretable. Among them, XGBoost is an important machine learning model, which is a boosting ensemble learning and a powerful classifier composed of a large number of Classification and Regression Trees (CART).

[4] The data is published on the official website of CBIRC: http://www.cbrc.gov.cn/chinese/newShouDoc/CDF5FDDEDAE14EFEB351CD93140E6554.html

CART regression tree will continuously be split along the binary tree according to features. For example, if current tree node J will be split based on the number of features - a, then samples with features less than b are split into left subtree, while samples with features greater than y are split into right subtree:

$$J_1(a,b) = \{X|X^{(a)} \le b\} \text{ (left subtree)}$$
$$J_2(a,b) = \{X|X^{(a)} > b\} \text{ (right subtree)}$$

In essence, CART regression tree is to divide the sample space by the feature dimension. The objective function generated by typical CART regression tree is:

$$\sum_{x_i \in J_m} (y_i - f(x_i))^2$$

Back to XGBoost, its core idea is to generate new split trees through continuous feature splitting. Each tree added is actually learning a new function to fit the residual of the last forecasting. Therefore, the objective function of XGBoost can be defined as:

$$y_i = \sum_j q_j x_{ij}$$

When there are k samples, the forecasting result of model at the n-th round is:

$$y_i^{(n)} = \sum_{k=1}^{n} f_k(x_i) = y_i^{(n-1)} + f_n(x_i)$$

Compared with GBDT and other machine learning technologies, XGBoost has the following benefits:

- XGBoost supports parallel computing and can make full use of the multi-thread capability of processor. Especially when running on Intel® platform, it can effectively utilize the powerful parallel computing capability brought by the latest instruction set such as Intel® AXV-512.
- XGBoost introduces a regularization item in its cost function, which can effectively control the complexity of the model and prevent the model from over-fitting.
- XGBoost supports column subsampling, which can not only prevent over-fitting, but also reduce computational complexity.
- GBDT uses only the first derivative information in its optimization, while XGBoost uses the first and second derivatives in the second Taylor expansion of the cost function, which has better forecasting effect.

Therefore, XGBoost machine learning model has been widely applied to credit risk forecasting solutions. The classical loan risk forecasting process when using XGBoost is shown in Figure 2-2-1, which is divided into several key steps such as data import, data cleaning and preparation, model building, model evaluation and model effect comparison:

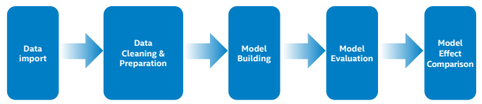


Figure 2-2-1 Forecasting Loan Risk Using XGBoost

■ RNN/LSTM-based Deep Learning Approach

Deep learning technologies such as RNN, a classic deep learning model, are also widely used in credit risk forecasting. A typical RNN structure will give an output for each input combined with the state of the current model. In addition to X from the input layer, the input of RNN's main structure A also has a cycle to provide the state at the current moment. At the same time, the state of A will be transferred from the current step to the next step.

LSTM is an important variant model of RNN, which can avoid the long-term dependence problem in the classical RNN structure by designing special "gate" structures, thereby greatly improving the memory duration. LSTM-based deep learning technology is very suitable for sequence-based analysis. In other words, banks can use a series of post-loan characteristics such as business operation and cash flow of enterprises, to predict the overdue loan risks that loans may face in the future.

However, the standalone deep learning approach is not ideal when it comes to the interpretability of processes, but financial institutions such as banks often need to interpret the results from inference, which means that they need to understand what information and conditions the model uses to obtain a specific forecast result. This interpretation can provide guidance for financial customers in improving business processes, customer experience, and more. The deep learning technologies often work like a black box, which makes interpretable deep learning technologies an important focus for future optimization.

■ Integrated Machine Learning and Deep Learning Solution

In order to improve interpretability and accuracy, it is necessary to adopt other technologies. Model stacking is a very effective technologies, which can improve the accuracy of regression or classification in most machine learning tasks, and can directly use the result files of different models for stacking, or can use the forecasting results of one model as the features of another model for training, and then obtain new forecasting results. Different types of models have different principles of learning and training, and the knowledge they have learned is also different, so stacking these models can improve the training effect. For example, the tree model XGBoost and LSTM deep learning model can be stacked to further enhance the forecasting capability and make the model interpretable. The overall structure of model stacking is as follows:

Stacking of Machine Learning and Deep Learning

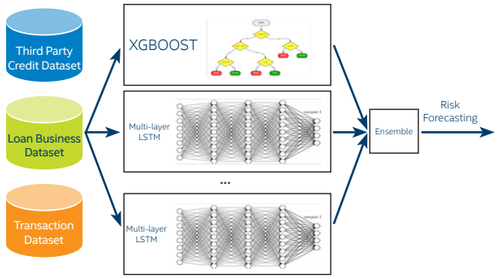


Figure 2-2-2 Overall Structure of Model Stacking

Algorithm Implementation of Credit Risk Forecasting Model

Training Data for Credit Risk Forecasting Model

The training data of credit risk forecasting model normally includes customers' loan transaction data in a few years and the quantitative data of customers' own operating conditions in the current month. In addition, the business logic of manual judgment will also be added to the dataset for training as an advanced feature.

■ Software Platform for Credit Risk Forecasting Model

The Distributed Machine Learning Community (DMLC) has now released an XGBoost open source package that is based on Intel® Distribution for Python and built for Intel® Optimization for TensorFlow Deep Learning Framework. The XGBoost open source package provides a wrapper class that allows the model to be used in conjunction with other classifiers or regressions in the Scikit-Learn framework.

XGBoost can speed up the training and inference process by using Intel® Distribution for Python. Intel® Distribution for Python has a built-in Intel® Data Analytics Acceleration Library (Intel® DAAL) for data analysis and machine learning, which can accelerate the machine learning process and make full use of the hardware resources of Intel architecture.

■ Intel® Optimization for TensorFlow

As a leading deep learning framework, TensorFlow has been widely used in AI applications in different industries. How to make the TensorFlow framework perform best on Intel® platforms is the focus of Intel® Optimization for TensorFlow. Its optimizations mainly include:

1) Integration of Intel® MKL-DNN.

2) Computational graph optimization.

3) Kernel optimization.

With the aid of above optimizations, it can be ensured that the most commonly used operations can be performed on the optimized MKL-DNN primitive, and the computational graph can be optimized by the way of operator fusion. In addition, multiple thread libraries can also be optimized so that they can coexist instead of competing for processor resources. These optimizations at software level significantly improved the overall training and inference performance without changing the neural network model. Please refer to "XGBoost Model and Training" below for details.

*** For more technical details about Intel® MKL-DNN, please refer to relevant content in the Technologies section of this guide.**

■ Data Pre-processing and Feature Engineering

According to specific tasks and data, data pre-processing and feature engineering will perform different processing, such as one-hot coding and data standardization. One-hot coding can convert categorical variables into numerical variables, while the data standardization is mainly to accelerate the training and convergence speed of model.

```
test_df = pd.concat([test_df,pd.get_dummies(test_df[Feature 1], prefix='F1')],axis=1)
test_df = pd.concat([test_df,pd.get_dummies(test_df['Feature 2'], prefix='F2')],axis=1)
... ...
test_df = pd.concat([test_df,pd.get_dummies(test_df[Feature N'], prefix='FN')],axis=1)
test_df.drop(['Feature 1'], axis=1, inplace=True)
test_df.drop([Feature 2], axis=1, inplace=True)
... ...
test_df.drop(['Feature N'], axis=1, inplace=True)
```

After converting categorical variables into numerical variables, the original categorical variables will be deleted and the entire dataset will be standardized:

```
std_scale = preprocessing.StandardScaler()
for column in need_to_scale:
test_df[column] = std_scale.fit_transform(test_df[column].reshape(-1, 1))
print("{} mean = {}, var = {} ".format(column, std_scale.mean_, std_scale.var_))
```

■ XGBoost Model and Training

■ XGBoost Model Implementation

The XGBoost model can be deployed directly by using XGBoost open source package or by using native XGBoost wrapper classes, the latter of which can adopt SKlearn style programming and is simpler and easier to use.

```
# fit model on training data
 eval_set = [(X_test, y_test)]
 # hyper parameters
h_param = {'learning_rate': 0.1,
        'n_estimators': 2000,
        'max_depth': 2,
        'min_child_weight': 10,
        'gamma': 0.0,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
        'objective': 'binary:logistic',
        'eval_metric': 'auc',
        'scale_pos_weight': imbalance_weight,
        'reg_lambda': 0.7,
        'reg_alpha': 1.25,
        'cv': 10}
model1 = XGBClassifier(learning_rate=h_param['learning_rate'],
                n_estimators=h_param['n_estimators'],
                max_depth=h_param['max_depth'],
                min_child_weight=h_param['min_child_weight'],
                gamma=h_param['gamma'],
                subsample=h_param['subsample'],
                colsample_bytree=h_param['colsample_bytree'],
                objective=h_param['objective'],
                eval_metric=h_param['eval_metric'],
                scale_pos_weight=h_param['scale_pos_weight'],
                reg_lambda=h_param['reg_lambda'],
```

```
                reg_alpha=h_param['reg_alpha'],
                seed=randint(1, 65535),
                cv=h_param['cv'])
xgb_param = model1.get_xgb_params()
print("-------Parameters------")
print(xgb_param)
 # Fit the algorithm on the data
model1.fit(X_train, y_train, eval_metric='auc')
```

■ XGBoost Parameter Tuning

XGBoost has so many hyper-parameters, and all parameters can be divided into the following three categories:

1) General parameters: used to macro function control and not required to adjust normally.

2) Booster parameters: Booster-related parameters that control each step, which need to be adjusted carefully and will affect the final performance.

3) Learning objective parameters: used to control the performance of training objectives, which are generally determined by tasks and normally do not need to be adjusted. Therefore, the parameters that need to be adjusted are mainly Booster-related parameters, as shown in the following table:

Parameter	Description
max_depth	The maximum depth of the tree. The deeper the tree gets, the more complex the model becomes, which can more easily result in over-fitting of the model
learning_rate	Learning rate or shrinkage factor
n_estimators	Number of weak classifiers
gamma	The minimum loss function drop value required for node splitting
min_child_weight	The minimum sum of sample weight (hessian) required by leaf node
subsample	Proportion of samples used to construct each tree
colsample_bytree	Proportion of features used to construct each tree
colsample_bylevel	Proportion of features used by the tree for each split at each level
reg_alpha	L1/L0 regular penalty coefficient
reg_lambda	L2 regular penalty coefficient

The Booster parameters listed in the above table can be adjusted step by step by using grid search. Grid search needs to select the optimal parameters through cross validation, which is very time-consuming when multiple parameters are optimized at the same time, so it is necessary to optimize the relevant parameters one by one or group by group. When adjusting, the number of trees can be adjusted by using the cv function provided by XGBoost, while other parameters can be adjusted by using XGBRegressor or XGBClassifier (in the XGBoost Sklearn package, the adjustment policy for regression and classification can follow the same rule) and GridSearchCV.

Next, define a function to adjust the number of optimal trees first:

```
def modelFit(alg, X_train, y_train, folds, useTrain=True, early_stopping_rounds=50):
    if useTrain:
        dtrain = xgb.DMatrix(data=X_train, label=y_train)
        params = alg.get_xgb_params()
        cvResult = xgb.cv(params=params, dtrain=dtrain, num_boost_round=params["n_estimators"], folds=folds,
                early_stopping_rounds=early_stopping_rounds, metrics="rmse")
        alg.set_params(n_estimators=cvResult.shape[0])
        print("n_estimators is: {}".format(cvResult.shape[0]))
    alg.fit(X_train, y_train)
    prediction = alg.predict(X_train)
    error = mean_squared_error(y_train, prediction)
    print("mean squared error is {}".format(error))
    fealmp = pd.Series(alg.get_booster().get_fscore()).sort_values(ascending=True)
    fealmp.plot(kind="barh", title="Feature Importance")
```

Step 2: Define a basic XGBRegressor (you can list all parameters that need to be adjusted for later use), and get the number of trees by invoking the function defined above:

```
xgbRegressor = XGBRegressor(
    n_estimators=2000,
    learning_rate=0.1,
    max_depth=6,
    min_child_weight=1,
    gamma=0,
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
Kfold = KFold(n_splits=5, random_state=0, shuffle=True)
modelFit(xgbRegressor, X_train, y_train, kfold)
```

The result is:

```
n_estimators is: {523}
```

Step 3: Adjust in groups according to the meaning of each parameter. At first, the step size of parameter adjustment can be enlarged for coarse adjustment. After the result is obtained, the step size can be reduced for fine adjustment:

```
param_grid1 = {"max_depth":range(3,10,2), "min_child_weight":range(1,10,2)}
xgbRegressor1 = XGBRegressor(
    n_estimators=523,
    learning_rate=0.1,
    max_depth=6,
    min_child_weight=1,
    gamma=0,
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
gridcv1 = GridSearchCV(xgbRegressor1, param_grid=param_grid1, scoring="neg_mean_squared_error", cv=5, n_jobs=-1)
gridcv1.fit(X_train, y_train)
gridcv1.best_params_, gridcv1.best_score_
```

The result is:

```
({'max_depth':9,'min_child_weight':5})
```

```
param_grid2 = {"max_depth":[8,9,10], "min_child_weight":[4,5,6]}
xgbRegressor2 = XGBRegressor(
    n_estimators=523,
    learning_rate=0.1,
    max_depth=6,
    min_child_weight=1,
    gamma=0,
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
gridcv2 = GridSearchCV(xgbRegressor2, param_grid=param_grid2, scoring="neg_mean_squared_error", cv=5, n_jobs=-1)
gridcv2.fit(X_train, y_train)
gridcv2.best_params_, gridcv2.best_score_
```

The result is:

```
({'max_depth':9,'min_child_weight':6})
```

Adjust the parameters to be optimized one by one to obtain the final optimal parameter: gamma

```
param_grid4 = {"gamma":[x/10 for x in range(0,6)]}
xgbRegressor4 = XGBRegressor(
    n_estimators=473,
    learning_rate=0.1,
    max_depth=9,
    min_child_weight=6,
    gamma=0,
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
gridcv4 = GridSearchCV(xgbRegressor4, param_grid=param_grid4, scoring="neg_mean_squared_error", cv=5, n_jobs=-1)
gridcv4.fit(X_train, y_train)
gridcv4.best_params_, gridcv4.best_score_
```

The result is:

```
Out[20]:({'gamma':0.1})
```

subsample	Proportion of samples used to construct each tree
colsample_bytree	Proportion of features used to construct each tree
colsample_bylevel	Proportion of features used by the tree for each split at each level

```
param_grid5 = {"subsample":[x/100 for x in range(70,90,5)], "colsample_bytree":[x/100 for x in range(70,90,5)],
              "colsample_bylevel":[x/100 for x in range(70,90,5)]}
xgbRegressor5 = XGBRegressor(
    n_estimators=473,
    learning_rate=0.1,
    max_depth=9,
    min_child_weight=6,
    gamma=0.1,
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
gridcv5 = GridSearchCV(xgbRegressor5, param_grid=param_grid5, scoring="neg_mean_squared_error", cv=5, n_jobs=-1)
gridcv5.fit(X_train, y_train)
gridcv5.best_params_, gridcv5.best_score_
```

The result is:

```
Out[20]:({'colsample_bylevel':0.75,'colsample_bytree':0.85,'subsample':0.7})
```

The following two parameters can be adjusted together.

reg_alpha	L1/L0 regular penalty coefficient
reg_lambda	L2 regular penalty coefficient

```
param_grid7 = {"reg_alpha":[1e-3, 1e-2, 0.05, 1e-1, 0, 1], "reg_lambda":[1e-3, 1e-2, 0.05, 1e-1, 0, 1]}
xgbRegressor7 = XGBRegressor(
    n_estimators=429,
    learning_rate=0.1,
    max_depth=9,
    min_child_weight=6,
    gamma=0.1,
    subsample=0.7,
    colsample_bytree=0.75,
    colsample_bylevel=0.95,
    reg_alpha=0,
    reg_lambda=1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
gridcv7 = GridSearchCV(xgbRegressor7, param_grid=param_grid7, scoring="neg_mean_squared_error", cv=5, n_jobs=-1)
gridcv7.fit(X_train, y_train)
gridcv7.best_params_, gridcv7.best_score_
```

The result is:

```
Out[28]:({'reg_alpha':1,'reg_lambda':0.1})
```

Finally, adjust the number of trees with all optimized parameters.

```
xgbRegressor9 = XGBRegressor(
    n_estimators=2000,
    learning_rate=0.05,
    max_depth=9,
    min_child_weight=6,
    gamma=0,
    subsample=0.7,
    colsample_bytree=0.75,
    colsample_bylevel=0.95,
    reg_alpha=1,
    reg_lambda=0.1,
    objective='reg:linear',
    seed=123,
    n_jobs=-1)
modelFit(xgbRegressor9, X_train, y_train, kfold)
```

■ Model Training and Inference

After all hyper-parameters of the model are determined, the parameters of XGBooster can be specified to generate Booster for training the model, and the training effect of the model can be evaluated by corresponding evaluation criteria of different tasks.

```
model = XGBClassifier()
eval_set = [(X_test, y_test)]
model.fit(X_train, y_train)
# make predictions for test data
y_predictions = model.predict(X_test)
# evaluate predictions
accuracy = metrics.accuracy_score(y_test, y_predictions)
```

```
print("Accuracy: %.2f%%" % (accuracy * 100.0))
acc = metrics.accuracy_score(y_test,y_predictions)
recall = metrics.recall_score(y_test,y_predictions)
f1 = metrics.f1_score(y_test,y_predictions)
precision = metrics.precision_score(y_test,y_predictions)
print("ACC: %.4f" % (acc * 100.0))
print("Precesion: %.4f" % (precision * 100.0))
print("Recall: %.4f" % (recall * 100.0))
print("F1-score: %.4f" % (f1 * 100.0))
metrics.confusion_matrix(y_test,y_predictions)
```

■ Use Intel® Optimization for TensorFlow to Implement LSTM

Implementation Approach

Integrated with Intel® MKL-DNN, Intel® Optimization for TensorFlow can make full use of hardware resources on Intel® hardware, and speed up network models such as LSTM by using vectorization, parallelization, and various optimizations such as Intel® DL Boost (VNNI Instruction Set).

■ Define Model Architecture

The two-layer LSTM network can be used to construct the network structure, in which one layer of LSTM network consists of a basic LSTM layer plus a Dropout layer, and the output of LSTM will connect to a three-layer fully connected network.

```
def make_cell(lstm_size):
    lstm = tf.nn.rnn_cell.BasicLSTMCell(lstm_size, state_is_tuple=True)
    drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob_)
    return drop
def LSTM(x, weights, biases):
    multi_layer_cell = tf.nn.rnn_cell.MultiRNNCell([make_cell(num_hidden) for _ in range(2)],
                                                   state_is_tuple=True)
    outputs, state = tf.nn.dynamic_rnn(multi_layer_cell, x, dtype = tf.float32)
    outputs = tf.transpose(outputs, [1, 0, 2])
    # We only need the last output tensor to pass into a classifier
    fc_32 = tf.layers.dense(outputs[-1], 32, activation=tf.nn.relu, name='FC_32')
    fc_16 = tf.layers.dense(fc_32, 8, activation=tf.nn.relu, name='FC_16')
    logit = tf.layers.dense(fc_16, num_classes, name='logit')
    return logit
```

Define loss function and optimizer to minimize loss function:

```
logits = LSTM(X, weights, biases)
weighted_logits = tf.multiply(logits, class_weight)
prediction = tf.nn.softmax(weighted_logits)
# Define loss and optimizer
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=weighted_logits, labels=Y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

■ Model Training

According to the configuration of the training server, set the system configuration parameters for training:

```
# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()
conf = tf.ConfigProto(intra_op_parallelism_threads=56,
                      inter_op_parallelism_threads=2,
                      allow_soft_placement=True)
```

Train and validate the model (multiple parameters to be observed can be monitored by using Tensorboard), and save the trained model:

```
# Start training
with tf.Session(config=conf) as sess:
    saver = tf.train.Saver()
    # Create a summary to monitor cost tensor
    tf.summary.scalar("loss_op", loss_op)
    tf.summary.scalar("accuracy", accuracy)
    merged_summary_op = tf.summary.merge_all()
    summary_writer = tf.summary.FileWriter('./tf_logs_train', sess.graph)
    test_writer = tf.summary.FileWriter("./tf_logs_test",sess.graph)
    # Run the initializer
    sess.run(init)
    i = 1
    for epoch in range(training_epochs):
        for b in range(total_batches):
            offset = (b * batch_size) % (lab_tr.shape[0] - batch_size)
            batch_x = X_tr[offset:(offset + batch_size), :]
            batch_y = lab_tr[offset:(offset + batch_size), :]
            _, c, summary = sess.run([train_op, loss_op, merged_summary_op],feed_dict={X: batch_x, Y: batch_y, keep_prob_: 0.9})
            summary_writer.add_summary(summary, i)
            i=i+1
    print("Optimization Finished!")
    saver.save(sess,"checkpoints/loan-lstm.ckpt")
    #用验证集验证模型效果：
    validation_batches = (X_vld.shape[0]//batch_size)
    i = 0
    precision_score_arr = []
    recall_score_arr = []
    f1_score_arr = []
    # Calculate accuracy for validation set
    for b in range(validation_batches):
        offset = (b * batch_size) % (lab_vld.shape[0] - batch_size)
        batch_x = X_vld[offset:(offset + batch_size), :]
        batch_y = lab_vld[offset:(offset + batch_size), :]
        # Calculate batch loss and accuracy
        pred, pred_2, Y_1, loss, acc, summary_val = sess.run([prediction, pred_val, Y_, loss_op, accuracy, merged_summary_op], feed_dict={X: batch_x, Y: batch_y, keep_prob_: 1.0})
        test_writer.add_summary(summary_val, i)
        i=i+1
        if b % display_step == 0 or b == 1:
            print("Step " + str(b) + ", Minibatch Loss= " + \
                "{:.4f}".format(loss) + ", Validation Accuracy= " + \
                "{:.3f}".format(acc))
            precision_val = precision_score(Y_1, pred_2)
            recall_val = recall_score(Y_1, pred_2)
            f1_val = f1_score(Y_1, pred_2)
            precision_score_arr.append(precision_val)
            recall_score_arr.append(recall_val)
            f1_score_arr.append(f1_val)
            print("Precision: {}".format(precision_val))
            print ("Recall: {}".format(recall_val))
            print ("f1_score: {}".format(f1_val))
            print ("confusion_matrix")
            print (confusion_matrix(Y_1, pred_2))
            fpr, tpr, tresholds = roc_curve(Y_1, pred_2)
    print("Validation precision mean = {}".format(sum(precision_score_arr)/len(precision_score_arr)))
    print("Validation recall mean = {}".format(sum(recall_score_arr)/len(recall_score_arr)))
    print("Validation f1 mean = {}".format(sum(f1_score_arr)/len(f1_score_arr)))
```

■ Model Inference

In the inference stage of the model, before obtaining the final inference results of the model by using the test set data, the previously saved model needs to be loaded first:

```
with tf.Session(config=conf) as sess:
    precision_score_arr = []
    recall_score_arr = []
    f1_score_arr = []
    # Restore
    saver.restore(sess, tf.train.latest_checkpoint('checkpoints'))
    batch_size = 100
    test_batches = (test_x.shape[0]//batch_size)
    i = 0
    print("test_batches = {}".format(test_batches))
    # Calculate accuracy for validation set
    for b in range(test_batches):
        offset = (b * batch_size) % (test_y.shape[0] - batch_size)
        batch_x = test_x[offset:(offset + batch_size), :]
        batch_y = test_y[offset:(offset + batch_size), :]
        # Calculate batch loss and accuracy
        pred, pred_2, Y_1, acc, summary_val = sess.run
        ([prediction, pred_val, Y_, accuracy, merged_summary
        _op], feed_dict={X: batch_x, Y: batch_y, keep_prob_ : 1.0})
        print("Step " + str(b) + \
            ", Test Accuracy= " + \
            "{:.3f}".format(acc))
        print("Y_1.shape = {}".format(Y_1.shape))
        print("pred_2.shape = {}".format(pred_2.shape))
        print("pred.shape = {}".format(pred.shape))
        print("Y_1: {}".format(Y_1))
        print("pred_2: {}".format(pred_2))
        print("pred: {}".format(pred))
        precision_val = precision_score(Y_1, pred_2)
        recall_val = recall_score(Y_1, pred_2)
        f1_val = f1_score(Y_1, pred_2)
        precision_score_arr.append(precision_val)
        recall_score_arr.append(recall_val)
        f1_score_arr.append(f1_val)
        print("Precision: {}".format(precision_val))
        print ("Recall: {}".format(recall_val))
        print ("f1_score: {}".format(f1_val))
        print ("confusion_matrix")
        print (confusion_matrix(Y_1, pred_2))
        fpr, tpr, tresholds = roc_curve(Y_1, pred_2)
    print("Test precision mean = {}".format(sum(precision
    _score_arr)/len(precision_score_arr)))
    print("Test recall mean = {}".format(sum(recall_score
    _arr)/len(recall_score_arr)))
    print("Test f1 mean = {}".format(sum(f1_score_arr)
    /len(f1_score_arr)))
```

Hybrid Overdue Loan Risk Forecasting Model

■ Model Introduction

The Hybrid Overdue Loan Risk Forecasting Model built on LSTM and traditional machine learning combines the advantages of machine learning and deep learning, which not only ensures the accuracy of forecasting through deep learning, but also provides interpretability of forecasting through machine learning. At the same time, this hybrid model can also be optimized using advanced tools and products such as Intel® Optimization for TensorFlow and Intel® Distribution for Python. Therefore, it can provide efficient forecasting services for commercial banks and other financial institutions.

The basic structure and workflow of this model are shown in Figure 2-2-3. The first step is feature analysis and data pre-processing. In this step, data from financial institutions' own big data platforms or data provided by third parties (e.g. data from credit bureaus) will be processed in the system, which includes processing of missing data, processing of data range, processing of data imbalance and analysis of important features of data. At the same time, as the dataset increases and becomes more complex, the model can also use different pre-processing toolkits and new models to deal with various types of data input.

The second step is to use the deep learning model (LSTM) and the traditional machine learning model (XGBoost/RF) to train and infer the sample data respectively, and obtain respective results. Then, the hybrid model will weigh the results, update the weights and make forecasting.

The final step of the model is to re-import the forecasting results of this round into the entry point of the model, update the features and weights according to the forecasting results, and make the next round of forecasting.

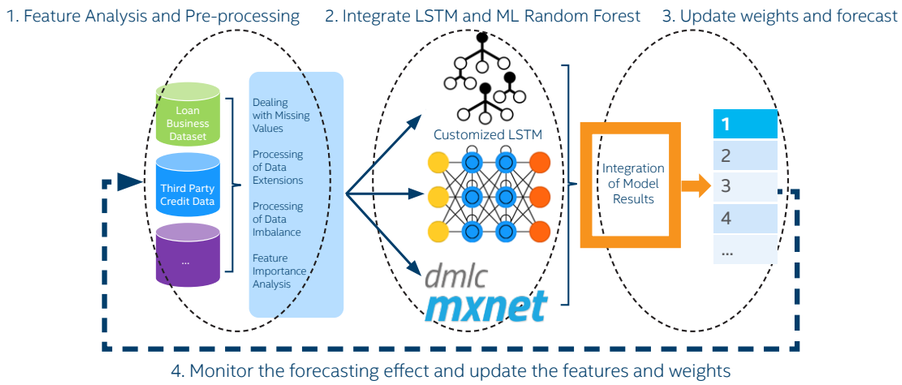


Figure 2-2-3 Hybrid Model Built on LSTM and Traditional Machine Learning

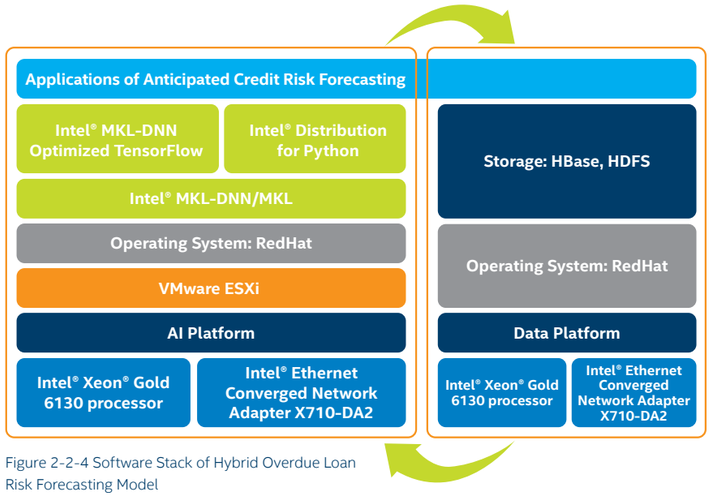


Figure 2-2-4 Software Stack of Hybrid Overdue Loan Risk Forecasting Model

■ Software Stack

The software stack of the Hybrid Overdue Loan Risk Forecasting Model is shown in Figure 2-2-4. On the left, the bottom layer is the hardware infrastructure built with Intel® Xeon® Gold 6130 Processor and Intel® Ethernet Converged Network Adapter X710-DA2. The next upper layer is an AI capability layer, in which a virtualized environment is provided by VMware ESXi, the RedHat Linux operating system (CentOS Linux release 7.4.1708) is installed, and Intel® MKL-DNN or Intel® MKL, Intel® Optimization for TensorFlow 1.10 and Intel® Distribution for Python are deployed. On the right, the bottom layer is the hardware infrastructure built with Intel® Xeon® Gold 6130 Processor and Intel® Ethernet Converged Network Adapter X710-DA2. The next upper layer is a data layer, in which the RedHat Linux operating system (CentOS Linux release 7.4.1708) is installed, and Apache HBase Distributed Database and Hadoop Distributed File System (HDFS) are deployed to provide read-write capability for distributed data storage. On top of AI capability layer and data layer, the hybrid overdue loan risk forecasting applications are deployed.

■ System Architecture

When building a real-world system, as shown in Figure 2-2-5, the hybrid overdue loan risk forecasting solution can be built in the following way. The whole system is divided into external data processing subsystem, online system and offline system from left to right. First, the system imports external data into the data planning and monitoring platform through the unified data interface, and then sends some data to the offline system through the service interface.

Secondly, in the offline system, part of the data from the external data subsystem and the online system will be imported into the Data Mart where the data will be cleaned, then these data will enter into the offline model training and algorithm deployment process, and then the trained model algorithm will be imported into the forecasting system of the online subsystem.

In the online subsystem, the front-end system will first push some data to the Data Mart for model training, while other data will be pushed by the data push system and enter into the distributed real-time computing system built by Storm cluster for forecasting and scheduling. After that, the data will enter into the forecasting system for inference and forecasting, and the results will be sent to the test platform for validation. The final results will go through the list management module and the risk label generation module, and then return to the algorithm deployment and model training module of the offline subsystem for algorithm iteration.

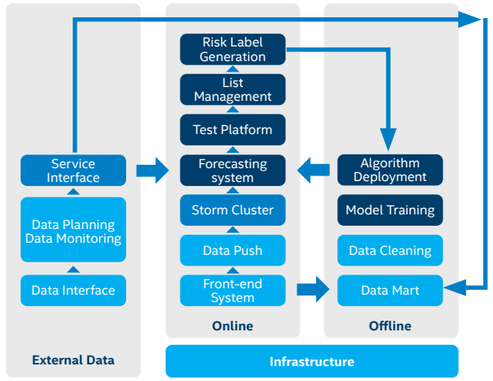


Figure 2-2-5 System Architecture of Hybrid Overdue Loan Risk Forecasting Solution

Recommended Hardware and Software Configuration

When building the above credit risk forecasting model, it can be configured by referring to the following environment built with Intel® platform:

Hardware Configuration

Name	Specification
Processor	Dual Intel® Xeon® Gold 6230 Processor or higher
Base Frequency	2.10GHz
Cores/Threads	16/32
HT	On
Turbo	On
Memory	192G (16G DDR4 2666MHz x12)
Hard Drive	Intel® DC S3320 SSD 480GB
BIOS	SE5C620.86B.00.01.0013.030920180427

Software Configuration

Name	Specification
Operating System	CentOS Linux release 7.4.1708 (Core)
Linux Kernel	3.10.0-957.12.1.el7.x86_64
Workloads	LSTM/XGBoost
Compiler	GCC 5.4
Library	Latest version of Intel® MKL
Framework	Intel® Optimization for TensorFlow Distribution
Other Software Configurations	Intel® Distribution for Python

Use Case: a Large Commercial Bank

Results

The actual deployment verification from the user shows that the final hybrid solution can effectively improve the forecasting accuracy and greatly reduce the forecasting delay. Specifically, compared with the manual forecasting solution, the accuracy of LSTM approach is doubled, while the forecasting accuracy of the hybrid model solution increases by more than 2 times and its forecasting delay is reduced to 2 days (the forecasting efficiency increases by more than 10 times). In addition, when using the online forecasting solution (loan risk forecasting), each forecasting time is less than 1 second, thereby significantly improving customer satisfaction.

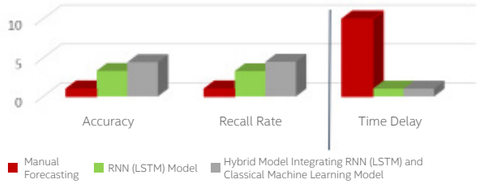


Figure 2-2-6 Comparison of Results of Different Forecasting Solutions

This commercial bank uses Intel® platform to build its hybrid model-based training and inference cluster. Intel® platform not only boasts high-performance cores and caches, but also can improve the performance of the framework with a number of hardware enhancements. For example, Intel® AVX-512 provided by Intel® Xeon® Processor and Intel® Xeon® Scalable Processor can provide excellent parallel computing capability.

Conclusion

Intel is committed to delivering customized solutions to financial users by combining its technical expertise with diversified AI technologies. The hybrid forecasting model based on LSTM and traditional machine learning not only achieves satisfactory results in forecasting accuracy, but also fully meets the user's requirements for interpretability of forecasting process.

Intel not only provides high-performance processors for this new hybrid forecasting model, but also provides a variety of software optimizations such as Intel® Optimization for TensorFlow and Intel® Distribution for Python, thus effectively improving the forecasting efficiency. At present, the hybrid forecasting model has been deployed in production environment in a commercial bank and has brought efficient and accurate credit risk forecasting capability to the user. In the future, both parties will work to use NLP models to analyze and forecast environmental data, and to provide more complete and accurate forecasting.

Although in the existing solution, the user adopts servers built with Intel® Xeon® Processor/Intel® Xeon® Scalable Processor. In the future, the user can also choose the 2nd Generation Intel® Xeon® Scalable Processor with better performance and more optimizations for AI to build its solutions.

AI-based Targeted Marketing Strategy for Financial Industry

Explore AI-based Targeted Marketing Strategy for Financial Industry

Background

For a long time, the financial industry has been an example of actively using AI to accelerate business development and improve marketing efficiency. The reasons are that: firstly, financial companies are often equipped with sophisticated IT systems and understand the importance of business data acquisition and accumulation, thereby accumulating huge amounts of data and providing a solid foundation for AI application; Secondly, banking, insurance, securities and other financial services all rely on data, and there are imperative needs for AI to empower the efficiency of a large number of tedious data processing tasks; In addition, the rapid development of deep learning has created more application scenarios for integrating AI into financial industry. Among these, the AI-based targeted marketing strategy for financial industry is receiving more and more attention.

The higher level of computerization and data advantages enable the financial enterprises to speed up the development of various recommender systems, and to implement important applications such as targeted marketing and personalized marketing by means of personalized recommendation and user profiling. By using massive structured/unstructured data, financial companies are building a series of marketing decision-making models in order to make in-depth analysis on the behavior preferences, user experience and purchase intentions of end users, and then speculate on the market prospects, thereby providing personalized suggestions for related financial products or business transactions and empowering marketing innovation of financial companies.

To embrace this trend, several financial companies chose to cooperate with Intel and built efficient recommender systems by relying on Analytics Zoo, an open source "Big Data Analytics + AI" platform of Intel and using Neural Collaborative Filtering (NCF), Wide and Deep (WAD) and other deep learning models.

Recommender System

■ Typical Recommender System

Recommender system (RS) is an information filtering tool for guiding users in a personalized way to discover their preferences from a large space of possible options. It is a critical tool to improve consumers' experience, promote marketing performance, and develop more accurate marketing products/plans. For example, if merchants can provide offers to consumers with the highest purchasing potential, this marketing campaign is undoubtedly very effective.

Now, the recommender system has become a key tool for many industries to promote sales and services. For example, when choosing movies to watch on Netflix, 80% of users will watch the movies recommended by the platform's recommender system[5]; Such figure on YouTube is 60%[6]; There is also data indicating that deep learning-based recommender systems are gaining wider acceptance in terms of recommendation quality[7].

Recommendation models can generally be divided into three categories, namely collaborative filtering, content-based and hybrid systems. The collaborative filtering-based recommendation algorithm believes that users will more likely choose products similar to those they have purchased, and makes recommendations by learning the historical interaction between the user and the products as well as utilizing explicit feedback (e.g., the user's previous rating) or implicit feedback (e.g., the user's comments after purchase). This algorithm does not need feature filtering and is more suitable as the initial model.

The content-based recommendation algorithm is based on the assumption that users usually like a product with similar content to the product they have been following. For example, if you have bought financial product A, and the content-based recommendation algorithm finds that financial product B has similar yield or investment years as the financial product A you bought before, then it will recommend financial product B to you. This algorithm can avoid the cold start problem of the recommender system, but its disadvantage is that it may repeatedly make the same recommendation. At the same time, this algorithm also relies on a large number of feature analysis.

At present, deep learning is increasingly being used to build highly efficient recommendation models. The traditional machine learning algorithms play a vital role in previous solutions, but with the increasing complexity of models and feature engineering, in recent years, people have also proposed many neural recommendation models based on deep learning to further improve the effectiveness of marketing campaign.

■ Process to Build a Recommender System

As shown in Figure 2-3-1, the building process of a recommender system can be composed of the following key steps: data cleaning, feature engineering, modeling, evaluation and optimization.

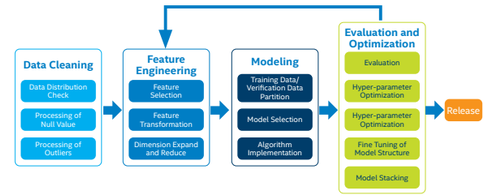


Figure 2-3-1 Process to Build a Recommender System

- In general, the raw data often contain all kinds of dirty data, which will greatly affect the accuracy of model training and forecasting. The data cleaning process is to re-examine and

[5] Carlos A Gomez-Uribe and Neil Hunt. 2016. The Recommender System of Netflix: Algorithms, Commercial Value and Innovation. ACM Trans. Manag. Inf. Syst. (TMIS) 6, 4 (2016), 13.

[6] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube Video Recommendation System. Proceedings of the Fourth ACM Conference on Recommender Systems Page 29 to Page 29 (RecSys '10).

[7] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning-based Recommender System: A Survey and New Perspectives. arXiv preprint arXiv:1707.07435, 2017.

check the data to ensure data consistency. Data cleaning mainly includes data distribution check, outlier processing, null value processing and other functions.

- The process of feature engineering is to extract useful information for forecasting from data and to transform dimensions to finally form feature vectors. This process includes key functions such as feature selection, feature transformation and dimension expand/reduce.
- The modeling process includes model selection, model training and algorithm implementation.
- Evaluation and Optimization include hyper-parameter optimization, model structure optimization, cross validation, and model stacking. After optimization, it is necessary to determine whether to return to the initial data cleaning and feature engineering according to the optimization results.

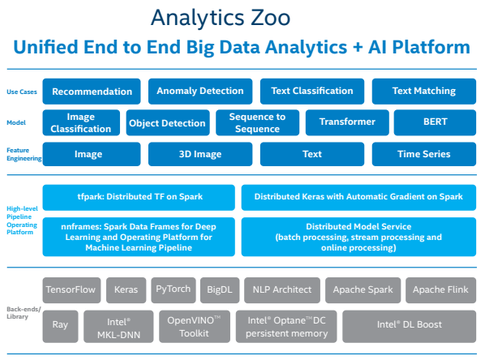
Analytics Zoo

Analytics Zoo is an Intel open source and unified "Big Data Analytics + AI" platform which can seamlessly integrate software and frameworks such as TensorFlow, Keras, PyTorch, BigDL, Ray, Spark and Flink into a unified system, and scale to large Apache Hadoop/Spark clusters to perform distributed training or forecasting required by deep learning.

Analytic Zoo can run on clusters built on Intel® Xeon® Scalable Processor to meet the requirements for enterprise deep learning. It allows users to develop and run deep learning applications directly on existing big data infrastructures, such as Apache Hadoop/Spark. Analytic Zoo can be seamlessly integrated into Web services (such as Spark Streaming and Kafka) by loading APIs using Plain Old Java Object (POJO), local Java API or Scala/Python models.

Analytics Zoo allows users to:

- Use Spark to process and analyze data.
- Use TensorFlow, Keras or PyTorch to develop deep learning models.
- Perform distributed training/inference on Spark and BigDL.



https://github.com/intel-analytics/analytics-zoo

Figure 2-3-2 Analytics Zoo provides rich end-to-end analytics capabilities and AI support

At the same time, Analytics Zoo also provides rich end-to-end analytics capabilities and AI support, including:

- Easy-to-use advanced analytics pipeline API (such as transfer learning support, automatic programming operation, Spark DataFrame, MLPipelines and online model service API).
- Common feature engineering operations for images, text, 3D images, and more.
- A large number of built-in deep learning models (e.g. object detection, image classification, text classification, recommendation, anomaly detection, text matching, sequence to sequence).
- Rich reference cases (e.g. anomaly detection, sentiment analysis, fraud detection, image similarity).

With Analytics Zoo, enterprise users can enjoy the following benefits:

- Analyze a large amount of data stored on the same big data cluster (such as HDFS, Apache HBase and Apache Hive), instead of moving or copying data.
- Add deep learning to existing analytics applications and machine learning pipelines instead of rebuilding them.
- Utilize existing big data clusters and infrastructure (resource allocation, load management and enterprise-level monitoring).
- When performing cross validation in the training phase, the deep learning algorithm will generate exponentially growing hidden embedded features and automatically perform internal feature selection and optimization, thus significantly reducing the workload of feature engineering. When building the model, the algorithm only pays attention to some predefined sliding features and custom overlapping features, and deletes most Long Time Variable (LTV) pre-computing task, which can save a lot of time and resources.
- Traditional machine learning (ML) methods rely heavily on human-machine learning experts to optimize models, while Analytics Zoo provides more options to find a better and more robust execution configuration, greatly improving the ability of automatic model optimization.
- Because Analytic Zoo can run as a standard Spark program on Intel® Xeon® processors, the deployment or operating costs are zero.

* For more technical details about Analytic Zoo, please refer to relevant content in the Technologies section of this guide.

Several Typical Deep Learning Models for AI Recommendation

■ Neural Collaborative Filtering (NCF) Model

NCF[8] model is one of the common deep learning-based recommendation algorithms[9]. As mentioned earlier, the collaborative filtering algorithm relies on explicit feedback and implicit feedback. However, in practice, explicit feedback is usually not available, while the implicit feedback is more common. For implicit feedback data, the Matrix Factorization (MF) can be used to abstract it into a recommendation problem. However, the traditional MF model, as a linear model of latent factor, can reflect the interaction between users and commodities, but it cannot actually reflect whether users like the commodities.

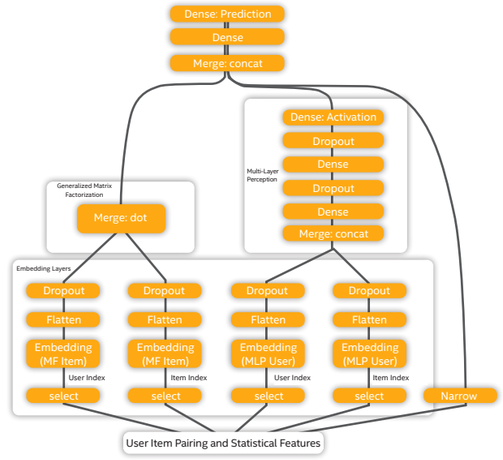


Figure 2-3-3 Sample of a Neural Collaborative Filtering (NCF) Model

NCF solves this problem by introducing a deep neural network. It can use a deep neural network (DNN) to learn interaction functions from data, thus eliminating the limitation of MF model. As shown in Figure 2-3-3, it uses Embedding Layer to map the sparse representation of the input layer to a new potential vector, and then sends user input and commodity input into the multi-Layer perception structure respectively. On the left, the model uses a Generalized Matrix Factorization (GMF) structure to handle linear interactions. On the right, the multi-Layer perception (MLP) is used to deal with nonlinear interactions. Finally, the two are merged to obtain better recommendation effect. Now, with Analytics Zoo, users can easily build NCF models.

■ Wide and Deep Model

The wide and deep learning model is a DNN-Linear hybrid model proposed in 2016, and is consisted of two parts: a wide component model and a deep component model. As shown in the right part of the model in Figure 2-3-4, the wide component is a single-layer perceptron, which works as a generalized linear model. Compared with the traditional recommender system which uses linear algorithm based on discrete features to make recommendations, the wide component model obtains the historical behavior data of users, such as which pages have been clicked and which goods have been purchased, and then forms discrete features through coding and carries out calculation.

The recommendation method adopting the wide component model has good effect on large-scale sparse data, and provides better interpretability for the model. Taking logistics regression (LR) as an example, each discrete feature can correspond to a weight value in the model, and the weight value of a feature is closely related to the effect of the feature on the result. However, the feature derivation of the wide component model requires a lot of human intervention and expert experience, and its forecasting effect is poor.

The deep component model is a multilayer perceptron similar to the neural collaborative filtering model. It obtains a series of vectors through deep learning and uses these vectors as part of the features to participate in training. The features generated by the deep component model have the following advantages: on the one hand, it can make up for the dimension limitation caused by artificially extracting features and improve forecasting accuracy, On the other hand, its features are automatically generated by the deep learning framework without human intervention, which can improve training efficiency. However, the vector generated by the deep component model is an implicit feature, which makes it hard to interpret the forecasting process.

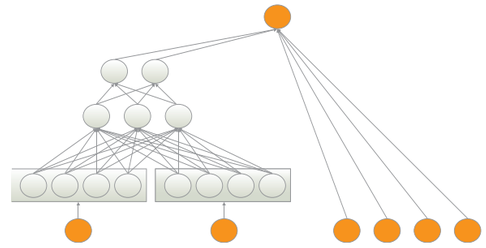


Figure 2-3-4 Diagram of Wide and Deep Model

Therefore, WAD model combines the wide component model and the deep component model together to obtain a more efficient recommender system. WAD model uses a SparseTensor and quite a few layers explicitly designed for sparse data calculation, e.g., SparseLinear, SparseJoinTable.

Analytics Zoo supports WAD model and provides DataFrame and Resilient Distributed Datasets (RDD) interfaces for data preparation and training, providing flexibility for different scenarios and allowing compatibility across Spark 1.5 to the latest versions.

Implementation of Analytics Zoo-based Model

■ System Implementation of Neural Collaborative Filtering Model

The following sections will explain how to build an explicit feedback-based NCF on Spark-based Analytics Zoo and BigDL.

The system environment is as follows:

- Python 2.7/3.5/3.6
- JDK 8
- Spark 1.6.0/2.1.1/2.1.2/2.2.0 (consistent with the Spark version for compiling Analytics Zoo)
- Analytics Zoo 0.5.0
- Jupyter Notebook 4.1

[8] For description about NCF technique, please refer to: https://www.comp.nus.edu.sg/~xiangnan/papers/ncf.pdf
[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 173–182.

■ Download or Install Analytics Zoo

For installing Analytics Zoo by using pip or downloading prebuilt package, please refer to: **https://analytics-zoo.github.io/master/#PythonUserGuide/install/** (you can also select User Guide –> Python –> Install in the left navigation menu on **https://analytics-zoo.github.io** homepage)

Run after installation via PIP

Users can easily run the sample using the following command:

```
export SPARK_DRIVER_MEMORY=22g
jupyter notebook --notebook-dir=./ --ip=* --no-browser
```

Please refer to the following URL for more instructions on running after PIP installation:
https://analytics-zoo.github.io/master/#PythonUserGuide/run/#run-after-pip-install

Run after installation via prebuilt package

Run the following command on Spark in local mode (master=local[*]) or cluster mode:

```
export SPARK_HOME=the root directory of Spark
export ANALYTICS_ZOO_HOME=the folder where you extract the downloaded Analytics Zoo zip package
  ${ANALYTICS_ZOO_HOME}/bin/jupyter-with-zoo.sh \
  --master ${MASTER} \
  --driver-cores 4  \
  --driver-memory 22g  \
  --total-executor-cores 4  \
  --executor-cores 4  \
  --executor-memory 22g
```

Please refer to the following URL for more instructions on running after non-PIP installation:
https://analytics-zoo.github.io/master/#PythonUserGuide/run/#run-without-pip-install

■ Implementation of Explicit Feedback-based NCF

The following will describe how to build a neural network recommender system and an explicit feedback-based NCF. The API of the recommender system can be used to build the model in Analytics Zoo, and the corresponding optimizer can be used to train the model.

The system (recommender system: principles, methods and evaluation[10]) usually prompts users to rate items on the system interface in order to build and improve the model. The accuracy of the recommendation depends on the number of ratings provided by the user.

NCF uses multilayer perceptron to learn user-item interaction function, and can express and generalize matrix factorization under its framework. includeMF (Boolean Type) is provided for users to build an NCF with or without matrix factorization.

The dataset used in this case is movieens-1M[11], which includes 1 million ratings (which are made on a 5-star scale) from 6,000 users on 4,000 movies. We will try to divide each pair (user and movies) into 5 categories and use the average absolute error to evaluate the effect of the algorithm.

References:

- A Keras implementation of Movie Recommendation; For details, please refer to: https://github.com/ririw/ririw.github.io/blob/master/assets/Recommending movies.ipynb and http://blog.richardweiss.org/2016/09/25/movie-embeddings.html
- NCF related papers; For details, please refer to: https://www.comp.nus.edu.sg/~xiangnan/papers/ncf.pdf

Import the necessary libraries:

```
In [1]:
from zoo.pipeline.api.keras.layers import *
from zoo.models.recommendation import UserItemFeature
from zoo.models.recommendation import NeuralCF
from zoo.common.nncontext import init_nncontext
import matplotlib
from sklearn import metrics
from operator import itemgetter
from bigdl.dataset import movielens
from bigdl.util.common import *
matplotlib.use('agg')
import matplotlib.pyplot as plt
%pylab inline
Populating the interactive namespace from numpy and matplotlib
```

Initialize NN context, then you can obtain a SparkContext for optimizing BigDL performance configuration:

```
In [2]:
sc = init_nncontext("NCF Example")
```

■ Data Preparation: Download and Read movielens Data of 1MB Size:

```
In [3]:
movielens_data = movielens.get_id_ratings("/tmp/movielens/")
```

The format of each record in the data is (userid, movieid, rating_score). The user ID ranges from 1 to 6,040 and the movie ID ranges from 1 to 3,952. The rating is made on a 5-star scale (only ratings with stars). Take a note of the number of users and movies for later use.

```
In [4]:
min_user_id = np.min(movielens_data[:,0])
max_user_id = np.max(movielens_data[:,0])
min_movie_id = np.min(movielens_data[:,1])
max_movie_id = np.max(movielens_data[:,1])
rating_labels= np.unique(movielens_data[:,2])
print(movielens_data.shape)
print(min_user_id, max_user_id, min_movie_id, max_movie_id, rating_labels)
(1000209, 3)
(1, 6040, 1, 3952, array([1, 2, 3, 4, 5]))
```

Convert the raw data into samples in RDD[12] format. In this example, BigDL's optimizer is directly used to train the model, which requires data to be provided in RDD format. In the following example, there is a BigDL data structure, which can be built using 2 numpy arrays, feature, and label respectively. The API interface here is Sample.from_ndarray(feature, label), which is used to convert labels from 1 to 0.

```
In [5]:
def build_sample(user_id, item_id, rating):
    sample = Sample.from_ndarray(np.array([user_id, item_id]), np.array([rating]))
    return UserItemFeature(user_id, item_id, sample)
pairFeatureRdds = sc.parallelize(movielens_data)\
    .map(lambda x: build_sample(x[0], x[1], x[2]-1))
pairFeatureRdds.take(3)
Out[5]:
[<zoo.models.recommendation.recommender.UserItemFeature at 0x11473ffd0>,
 <zoo.models.recommendation.recommender.UserItemFeature at 0x124eb9110>,
 <zoo.models.recommendation.recommender.UserItemFeature at 0x11473fed0>]
```

Randomly divide the data into sequence (80%) and validation (20%).

```
In [6]:
trainPairFeatureRdds, valPairFeatureRdds = pairFeatureRdds.randomSplit([0.8, 0.2], seed= 1)

valPairFeatureRdds.cache()
train_rdd= trainPairFeatureRdds.map(lambda pair_feature: pair_feature.sample)
val_rdd= valPairFeatureRdds.map(lambda pair_feature: pair_feature.sample)
val_rdd.persist()
Out[6]:
PythonRDD[3] at RDD at PythonRDD.scala:48
In [7]:
print(train_rdd.count())
train_rdd.take(3)
799923
Out[7]:
[Sample: features: [JTensor: storage: [  1. 661.], shape: [2], float], labels: [JTensor: storage: [2.], shape: [1], float],
 Sample: features: [JTensor: storage: [  1. 914.], shape: [2], float], labels: [JTensor: storage: [2.], shape: [1], float],
 Sample: features: [JTensor: storage: [1.000e+00 3.408e+03], shape: [2], float], labels: [JTensor: storage: [3.], shape: [1], float]]
```

■ Build the Model

In Analytics Zoo, you can easily invoke the Neuracf API to build an NCF model - you only need to specify the user count, item count, and category number according to the data, then add hidden layers as needed, and optionally include matrix factorization in the network. In this model, the optimizer of BigDL or NNClassifier of Analytics Zoo can be used. The following case demonstrates how to use BigDL's optimizer.

```
In [8]:
ncf = NeuralCF(user_count=max_user_id,
               item_count=max_movie_id,
               class_num=5,
               hidden_layers=[20, 10],
               include_mf = False)
creating: createZooKerasInput
creating: createZooKerasFlatten
creating: createZooKerasSelect
creating: createZooKerasFlatten
creating: createZooKerasSelect
creating: createZooKerasEmbedding
creating: createZooKerasEmbedding
creating: createZooKerasFlatten
creating: createZooKerasFlatten
creating: createZooKerasMerge
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasModel
creating: createZooNeuralCF
```

■ Compile the Model

Compile the model according to the specific optimizer, loss and evaluation criteria, and the optimizer will minimize the loss of the neural network relative to its weight/deviation on the training set. To create an optimizer in BigDL, the following parameters must be specified at least, including model (neural network model), criteria (lose function), traing_dd (training dataset), and batch size.

For more information on creating efficient optimizer, please refer to the following programming guide and optimizer manual.
Programming Guide: https://bigdl-project.github.io/master/#ProgrammingGuide/optimization/
Optimizer: https://bigdl-project.github.io/master/#APIGuide/Optimizers/Optimizer/

```
In [9]:
ncf.compile(optimizer= "adam",
            loss= "sparse_categorical_crossentropy",
            metrics=['accuracy'])
creating: createAdam
creating: createZooKerasSparseCategoricalCrossEntropy
creating: createZooKerasSparseCategoricalAccuracy
```

■ Collect Log

The summary can be viewed through the TensorBoard:

```
In [10]:
tmp_log_dir = create_tmp_path()
ncf.set_tensorboard(tmp_log_dir, "training_ncf")
```

■ Train the Model

```
In [11]:
ncf.fit(train_rdd,
        nb_epoch= 10,
        batch_size= 8000,
        validation_data=val_rdd)
```

■ Forecast

Analytics Zoo model uses the **model.predict (val-rdd)** API to infer based on given data. Returns the RDD result, and returns the forecasting label through the Predict_class class:

[10] For details, please refer to: http://www.sciencedirect.com/science/article/pii/S1110866515000341
[11] For details about the dataset, please refer to : https://grouplens.org/datasets/movielens/1m/
[12] For detailed description of RDD samples, please refer to: https://bigdl-project.github.io/master/#APIGuide/Data/#sample

```
In [12]:
results = ncf.predict(val_rdd)
results.take(5)

results_class = ncf.predict_class(val_rdd)
results_class.take(5)
Out[12]:
[5, 5, 4, 4, 4]
```

In Analytics Zoo, three unique API are provided to forecast user-item pairs and make recommendations for users or given candidate items:

```
In [13]:
userItemPairPrediction = ncf.predict_user_item_pair(valPairFeatureRdds)
for result in userItemPairPrediction.take(5): print(result)
UserItemPrediction [user_id: 1, item_id: 1193, prediction: 5, probability: 0.476881682873]
UserItemPrediction [user_id: 1, item_id: 2804, prediction: 5, probability: 0.451132953167]
UserItemPrediction [user_id: 1, item_id: 594, prediction: 4, probability: 0.481520324945]
UserItemPrediction [user_id: 1, item_id: 2398, prediction: 4, probability: 0.415099412203]
UserItemPrediction [user_id: 1, item_id: 1097, prediction: 4, probability: 0.453616738319]
```

Recommend 3 items for each user and give candidates in RDD features:

```
In [14]:
userRecs = ncf.recommend_for_user(valPairFeatureRdds, 3)
for result in userRecs.take(5): print(result)
UserItemPrediction [user_id: 4904, item_id: 2019, prediction: 5, probability: 0.9045779109]
UserItemPrediction [user_id: 4904, item_id: 318, prediction: 5, probability: 0.902075052261]
UserItemPrediction [user_id: 4904, item_id: 912, prediction: 5, probability: 0.866227447987]
UserItemPrediction [user_id: 3456, item_id: 1356, prediction: 5, probability: 0.832679390907]
UserItemPrediction [user_id: 3456, item_id: 1374, prediction: 5, probability: 0.799858570099]
```

Recommend 3 users for each item and give candidates in RDD features:

```
In [15]:
itemRecs = ncf.recommend_for_item(valPairFeatureRdds, 3)
for result in itemRecs.take(5): print(result)
UserItemPrediction [user_id: 195, item_id: 3456, prediction: 5, probability: 0.525387585163]
UserItemPrediction [user_id: 1926, item_id: 3456, prediction: 5, probability: 0.483191937208]
UserItemPrediction [user_id: 4298, item_id: 3456, prediction: 5, probability: 0.468448847532]
UserItemPrediction [user_id: 1271, item_id: 1080, prediction: 5, probability: 0.747303187847]
UserItemPrediction [user_id: 2447, item_id: 1080, prediction: 5, probability: 0.743132531643]
```

■ Evaluation

Draw training and validation loss curves:

```
In [16]:
#retrieve train and validation summary object and read the loss data into ndarray's.
train_loss = np.array(ncf.get_train_summary("Loss"))
val_loss = np.array(ncf.get_validation_summary("Loss"))
#plot the train and validation curves
# each event data is a tuple in form of (iteration_count, value, timestamp)
plt.figure(figsize = (12,6))
plt.plot(train_loss[:,0],train_loss[:,1],label='train loss')
plt.plot(val_loss[:,0],val_loss[:,1],label='val loss',color='green')
plt.scatter(val_loss[:,0],val_loss[:,1],color='green')
plt.legend();
plt.xlim(0,train_loss.shape[0]+10)
plt.grid(True)
plt.title("loss")
Out[16]:
Text(0.5,1,'loss')
```

Draw accuracy rate:

```
In [17]:
plt.figure(figsize = (12,6))
top1 = np.array(ncf.get_validation_summary("Top1Accuracy"))
plt.plot(top1[:,0],top1[:,1],label='top1')
plt.title("top1 accuracy")
plt.grid(True)
plt.legend();
```

Implementation Case of Wide and Deep Network

In the following section, the recommended API of Analytics Zoo will be used to build a wide linear model and a deep neural network, i.e. a wide and deep network, and BigDL optimizer will be used to train the network. The wide and deep model combines memory strength and generalization together, and can be used for general large-scale regression and classification problems, including burst input features (e.g., category features with a large number of possible feature values).

The system environment is as follows:

- Python 2.7/3.5/3.6
- DK 8
- Spark 1.6.0/2.1.1/2.1.2/2.2.0 (consistent with the Spark version for compiling Analytics Zoo)
- Analytics Zoo 0.5.0
- Jupyter Notebook 4.1

To download or install Analytics Zoo, please refer to how to run Analytics Zoo on page 38.

■ Initialization

Import required libraries

```
In [1]:
from zoo.models.recommendation import *
from zoo.models.recommendation.utils import *
from zoo.common.nncontext import init_nncontext
import os
import sys
import datetime as dt
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
%pylab inline
Populating the interactive namespace from numpy and matplotlib
```

Initialize NN context, then you can obtain a SparkContext for optimizing BigDL performance configuration:

```
In [2]:
sc = init_nncontext("WideAndDeep Example")
```

■ Data Preparation

Download and read 1MB movielens data, and learn about its dimensions:

```
In [3]:
from bigdl.dataset import movielens
movielens_data = movielens.get_id_ratings("/tmp/movielens/")
min_user_id = np.min(movielens_data[:,0])
max_user_id = np.max(movielens_data[:,0])
min_movie_id = np.min(movielens_data[:,1])
max_movie_id = np.max(movielens_data[:,1])
rating_labels= np.unique(movielens_data[:,2])
print(movielens_data.shape)
print(min_user_id, max_user_id, min_movie_id, max_movie_id, rating_labels)
(1000209, 3)
(1, 6040, 1, 3952, array([1, 2, 3, 4, 5]))
```

Convert rating data into data frames, and read user and item data as data frames. Convert label from 1 to 0:

```
In [4]:
sqlContext = SQLContext(sc)
from pyspark.sql.types import *
from pyspark.sql import Row
Rating = Row("userId", "itemId", "label")
User = Row("userId", "gender", "age" ,"occupation")
Item = Row("itemId", "title" ,"genres")

ratings = sc.parallelize(movielens_data)\
    .map(lambda l: (int(l[0]), int(l[1]), int(l[2])-1))\
    .map(lambda r: Rating(*r))
ratingDF = sqlContext.createDataFrame(ratings)

users= sc.textFile("/tmp/movielens/ml-1m/users.dat")\
    .map(lambda l: l.split("::")[0:4])\
    .map(lambda l: (int(l[0]), l[1], int(l[2]), int(l[3])))\
    .map(lambda r: User(*r))
userDF = sqlContext.createDataFrame(users)

items = sc.textFile("/tmp/movielens/ml-1m/movies.dat")\
    .map(lambda l: l.split("::")[0:3])\
    .map(lambda l: (int(l[0]), l[1], l[2].split('|')[0]))\
    .map(lambda r: Item(*r))
itemDF = sqlContext.createDataFrame(items)
```

Link and convert data. For example, gender will be used as the classification feature, while occupation and gender will be used as the cross feature:

```
In [5]:
from pyspark.sql.functions import col, udf

gender_udf = udf(lambda gender: categorical_from_vocab_list(gender, ["F", "M"], start=1))
bucket_cross_udf = udf(lambda feature1, feature2: hash_bucket(str(feature1) + "_" + str(feature2), bucket_size=100))
genres_list = ["Crime", "Romance", "Thriller", "Adventure", "Drama", "Children's",
               "War", "Documentary", "Fantasy", "Mystery", "Musical", "Animation", "Film-Noir", "Horror",
               "Western", "Comedy", "Action", "Sci-Fi"]
genres_udf = udf(lambda genres: categorical_from_vocab_list(genres, genres_list, start=1))

allDF = ratingDF.join(userDF, ["userId"]).join(itemDF, ["itemId"]) \
    .withColumn("gender", gender_udf(col("gender")).cast("int")) \
    .withColumn("age-gender", bucket_cross_udf(col("age"), col("gender")).cast("int")) \
    .withColumn("genres", genres_udf(col("genres")).cast("int"))
allDF.show(5)
+------+------+-----+------+---+----------+--------------+------+----------+
|itemId|userId|label|gender|age|occupation|         title|genres|age-gender|
+------+------+-----+------+---+----------+--------------+------+----------+
|    26|  3391|    3|     2| 18|         4|Othello (1995)|     5|        24|
|    26|  1447|    4|     2| 18|         4|Othello (1995)|     5|        24|
|    26|  5107|    3|     1| 45|         0|Othello (1995)|     5|         5|
|    26|  2878|    3|     1| 50|        20|Othello (1995)|     5|        47|
|    26|  1527|    1|     2| 18|        10|Othello (1995)|     5|        24|
+------+------+-----+------+---+----------+--------------+------+----------+
only showing top 5 rows
```

The above is the special data feature information shared by the wide and deep model and its feature generation. Here, we treat occupation and gender as a generalized basic part, age and gender as a generalized cross part, genre and gender as indicators, and user ID and item ID are used for embedding.

```
In [6]:
bucket_size = 100
column_info = ColumnFeatureInfo(
        wide_base_cols=["occupation", "gender"],
        wide_base_dims=[21, 3],
        wide_cross_cols=["age-gender"],
        wide_cross_dims=[bucket_size],
        indicator_cols=["genres", "gender"],
        indicator_dims=[19, 3],
        embed_cols=["userId", "itemId"],
        embed_in_dims=[max_user_id, max_movie_id],
        embed_out_dims=[64, 64],
        continuous_cols=["age"])
```

When converting data into RDD sample, BigDL's optimizer can be used directly to train the model, which requires data to be provided in RDD (sample) format. One example is a BigDL data structure, which can be built using 2 numpy arrays, feature, and label respectively. The API interface is sample. from ndarray (feature, label). The wide and deep model needs two input tensors, one being the sparse tensor of the wide model, and the other being the dense tensor of the deep model.

```
In [7]:
rdds = allDF.rdd\
.map(lambda row: to_user_item_feature(row, column_info))\
.repartition(4)
trainPairFeatureRdds, valPairFeatureRdds = rdds.randomSplit([0.8, 0.2], seed= 1)
valPairFeatureRdds.persist()
train_data= trainPairFeatureRdds.map(lambda pair_feature: pair_feature.sample)
test_data= valPairFeatureRdds.map(lambda pair_feature: pair_feature.sample)
```

■ Create the Wide and Deep Model

In Analytics Zoo, the wide and deep model can be easily built by invoking the **wide and deep** API. Only the column information of model type, category number and features needs to be specified according to the data, and other default parameters in the network, such as hidden layer, can also be changed. In this model, the optimizer of BigDL or NNClassifier of Analytics Zoo can be used. The following example demonstrates how to use BigDL's optimizer:

```
In [8]:
wide_n_deep = WideAndDeep(5, column_info, "wide_n_deep")
creating: createZooKerasInput
creating: createZooKerasInput
creating: createZooKerasInput
creating: createZooKerasInput
creating: createZooKerasSparseDense
creating: createZooKerasFlatten
creating: createZooKerasSelect
creating: createZooKerasEmbedding
creating: createZooKerasFlatten
creating: createZooKerasFlatten
creating: createZooKerasSelect
creating: createZooKerasEmbedding
creating: createZooKerasFlatten
```

```
creating: createZooKerasMerge
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasDense
creating: createZooKerasMerge
creating: createZooKerasActivation
creating: createZooKerasModel
creating: createZooWideAndDeep
```

■ Create and Optimize the Training Model:

```
In [9]:
wide_n_deep.compile(optimizer = "adam",
                    loss= "sparse_categorical_crossentropy",
                    metrics=['accuracy'])
creating: createAdam
creating: createZooKerasSparseCategoricalCrossEntropy
creating: createZooKerasSparseCategoricalAccuracy
In [10]:
tmp_log_dir = create_tmp_path()
wide_n_deep.set_tensorboard(tmp_log_dir, "training_wideanddeep")
```

Train the network until it is completed, and obtain a model which has completed training:

```
In [11]:
%%time
# Boot training process
wide_n_deep.fit(train_data,
                batch_size = 8000,
                nb_epoch = 10,
                validation_data = test_data)
print("Optimization Done.")
Optimization Done.
CPU times: user 54.3 ms, sys: 19.7 ms, total: 74 ms
Wall time: 2min 30s
```

■ Forecast and Recommend

Analytics Zoo model uses the **model.predict(val-rdd)** API to infer based on given data. Returns RDD results. The **Predict_class** class returns the forecasting label.

```
In [12]:
results = wide_n_deep.predict(test_data)
results.take(5)

results_class = wide_n_deep.predict_class(test_data)
results_class.take(5)
Out[12]:
[4, 2, 4, 5, 2]
```

In Analytics Zoo, three unique APIs are provided to forecast user-item pairs and make recommendations for users or given candidate items:

```
In [13]:
userItemPairPrediction = wide_n_deep.predict_user_item_pair(valPairFeatureRdds)
for result in userItemPairPrediction.take(5): print(result)
UserItemPrediction [user_id: 5305, item_id: 26, prediction: 4, probability: 0.447520256042]
UserItemPrediction [user_id: 1150, item_id: 26, prediction: 2, probability: 0.42147180438]
UserItemPrediction [user_id: 4294, item_id: 26, prediction: 4, probability: 0.338612318039]
UserItemPrediction [user_id: 5948, item_id: 26, prediction: 5, probability: 0.385789096355]
UserItemPrediction [user_id: 3825, item_id: 26, prediction: 2, probability: 0.292931675911]
```

Recommend 3 items for each user and give candidates in RDD features:

```
In [14]:
userRecs = wide_n_deep.recommend_for_user(valPairFeatureRdds, 3)
for result in userRecs.take(5): print(result)
UserItemPrediction [user_id: 4904, item_id: 1221, prediction: 5, probability: 0.901316523552]
UserItemPrediction [user_id: 4904, item_id: 593, prediction: 5, probability: 0.890776693821]
UserItemPrediction [user_id: 4904, item_id: 913, prediction: 5, probability: 0.888917982578]
UserItemPrediction [user_id: 1084, item_id: 50, prediction: 5, probability: 0.632001161575]
UserItemPrediction [user_id: 1084, item_id: 912, prediction: 5, probability: 0.584099054337]
```

Recommend 3 users for each item and give candidates in RDD features:

```
In [15]:
itemRecs = wide_n_deep.recommend_for_item(valPairFeatureRdds, 3)
for result in itemRecs.take(5): print(result)
UserItemPrediction [user_id: 1835, item_id: 1084, prediction: 5, probability: 0.745298802853]
UserItemPrediction [user_id: 3864, item_id: 1084, prediction: 5, probability: 0.744241654873]
UserItemPrediction [user_id: 5582, item_id: 1084, prediction: 5, probability: 0.739497065544]
UserItemPrediction [user_id: 4511, item_id: 3764, prediction: 4, probability: 0.44239372015]
UserItemPrediction [user_id: 116, item_id: 3764, prediction: 4, probability: 0.365347951651]
```

■ Draw Convergence Curve:

```
In [16]:
#retrieve train and validation summary object and read the loss data into ndarray's.
train_loss = np.array(wide_n_deep.get_train_summary("Loss"))
val_loss = np.array(wide_n_deep.get_validation_summary("Loss"))
#plot the train and validation curves
# each event data is a tuple in form of (iteration_count, value, timestamp)
plt.figure(figsize = (12,6))
plt.plot(train_loss[:,0],train_loss[:,1],label='train loss')
plt.plot(val_loss[:,0],val_loss[:,1],label='val loss',color='green')
plt.scatter(val_loss[:,0],val_loss[:,1],color='green')
plt.legend();
plt.xlim(0,train_loss.shape[0]+10)
plt.grid(True)
plt.title("loss")
Out[16]:
Text(0.5,1,'loss')
```

Draw Accuracy Curve:

```
In [17]:
plt.figure(figsize = (12,6))
top1 = np.array(wide_n_deep.get_validation_summary
("Top1Accuracy"))
plt.plot(top1[:,0],top1[:,1],label='top1')
plt.title("top1 accuracy")
plt.grid(True)
plt.legend();
plt.xlim(0,train_loss.shape[0]+10)
Out[17]:
(0, 1010)

In [18]:
valPairFeatureRdds.unpersist()
Out[18]:
PythonRDD[82] at RDD at PythonRDD.scala:48
In [19]:
sc.stop()
```

Recommended Hardware and Software Configuration

When building the above AI-based targeted marketing strategy, it can be configured by referring to the following environment built with Intel® platform:

Hardware Configuration

Name	Specification
Processor	Dual Intel® Xeon® Processor E5-2650 v4 or higher
Base Frequency	2.20GHz
Base Frequency	12/24
HT	BIOS Default Settings (enabled or disabled)
Turbo	BIOS Default Settings (enabled or disabled)
Memory	384G (32G DDR4 2666MHz x12)
Hard Drive	21TB
BIOS	Factory settings, or any later upgraded version
Other Hardware Configurations	10GbE Network Bandwidth

Software Configuration

Name	Specification
Operating System	Ubuntu 14.04 LTS * For latest supported operating system versions, please refer to **https://analytics-zoo.github.io/**
Linux Kernel	3.14
Workloads	Analytics Zoo based NCF, WAD, ALS model training & model inference.
Compiler	gcc 4.8
Library	Analytics Zoo- bigdl_0.6.0-spark_2.2.0 (Intel® MKL included) • Spark MLlib 2.2.0
Framework	Analytics Zoo, BigDL
Other Software Configurations	• Hadoop release: Cloudera Distributed Hadoop (CDH) 5.12.1 • Spark version: 2.2 • Java Platform, Standard Edition Development Kit (JDK) 1.8

Use Cases

China Life Insurance Shanghai Data Center Reimplementing Life Insurance Services

Background

As an important member of China Life Insurance, an ultra-large insurance company with premium income of more than 400 billion yuan, China Life Insurance Shanghai Data Center is trying to build advanced AI capabilities to allow its salesmen to efficiently recommend personalized insurance products to different customers, thereby solving the problems caused by the continuous expansion of business scale and insurance policies.

Previously, when making recommendations, salesmen could only rely on their own experience and choose the company's key insurance policies, with little consideration given to customers' own needs. This strategy leads to two problems. First, the customer's needs are not actually met. Secondly, customers may cancel their orders or policies, causing losses to the company's revenue.

The lack of good methodology is the main reason for this problem when salesmen recommend insurance products. Especially for inexperienced young salesmen, they are more likely to use misleading sales tactics. Therefore, China Life Insurance Shanghai Data Center plans to rely on their abundant data and build an AI-based recommendation model in order to support salesmen to promote customer satisfaction by recommending insurance products more efficiently.

■ Solution and Results

The platform architecture of China Life Insurance Shanghai Data Center's business recommender system is shown in Figure 2-3-5. The platform is built on Analytics Zoo, in which the big data platform adopts CDH version 5.10 and uses Sqoop to import data from business system into HDFS. Data cleaning and partial pre-processing are performed using Hive/ Impala. Data pre-processing can also be performed using Python and Scala, and then the processed data is stored in IMPALA or HIVE. Then, Spark On Hive is used to read the data in a structured form and BigDL is invoked for model training.

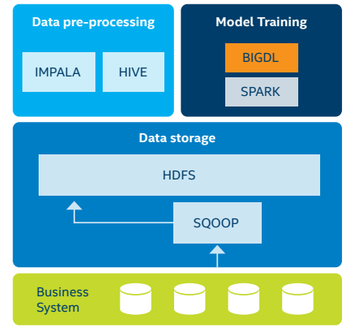


Figure 2-3-5 Platform Architecture of China Life Insurance Shanghai Data Center's Business Recommender System

The deep learning-based business recommender system of China Life Insurance Shanghai Data Center primarily adopts NCF model. As described in the introduction of NCF model above, the model is divided into left and right parts. The left part is the generalized matrix factorization, which performs multiplication on the input vectors. The model on the right is the multi-Layer perception, which concatenates the input features of user and item and performs multi-layer transformation. Then, the results of the two models are integrated in the upper layer, and converted into final user preference values by sigmoid method. In this case, the setting values of NCF parameters are as follow:

- Embedding is initialized to a normal distribution with a mean value of 0 and a variance of 0.01.
- Batch Size is set to 2800.
- The tuning method is Adam.

The output of the model is each user's rating for each insurance policy, and the top insurance policies with higher ratings are recommended to users by ranking the ratings in reverse order.

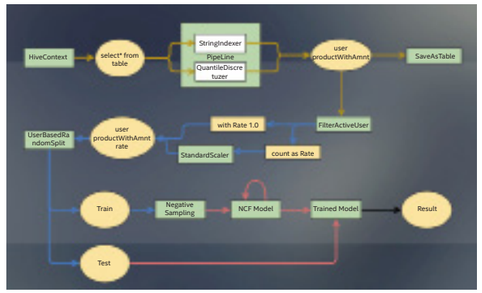


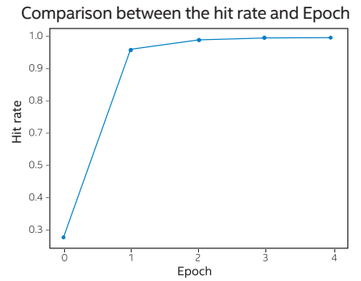
Figure 2-3-6 Basic Processing Flow of China Life Insurance Shanghai Data Center's Recommendation Model

As shown in Figure 2-3-6, the basic processing flow in the recommendation model consists of the following steps:

1. Pre-process the data by using HiveContext to directly read data from Hive in SQL mode.
2. After data is read, it will be stored as a DataFrame object in Spark; In order to make the data available for neural network, the data is mapped into discrete data by using String indexer through PipeLine interface in Spark.
3. After acquiring such dataset (user, productWithAmnt), deduplication will be performed, then a user purchase preference rating is added to each data, for example, the rating of a purchased item will be set to 1.
4. Divide training set and test set, and add negative data into training data through **Spark** API.
5. Finally, the training set will be used for data training and the test set will be used to validate the training results of the model.

By using these steps, the platform built on Analytics Zoo will provide strong support for Spark DataFrame, MLPipelines, and more by using a variety of advanced analysis pipeline APIs and features provided by Analytics Zoo, thus effectively improving the work efficiency of the whole process.

China Life Insurance Shanghai Data Center evaluated the effect of its recommender system by using two key indicators, namely Hit Rate and Normalized Distributed Cumulative Gain (NDCG). In this case, as shown in Figure 2-3-7, the Hit Rate of China Life Insurance Shanghai Data Center's recommender system is 99.8%, and the NDCG reaches 0.66, both of which exceed their expectation. Therefore, it can be said that the recommender system performs really well.



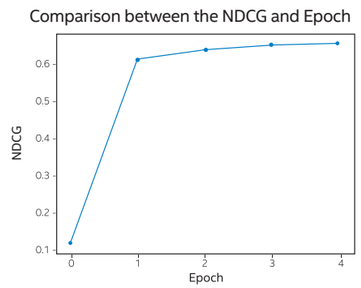


Figure 2-3-7 Evaluation Results of China Life Insurance Shanghai Data Center's Recommender System

Optimization of MasterCard Recommender Service

■ Background and Challenges

As a leading global provider of payment solutions, MasterCard has issued 2.6 billion credit cards with an annual transaction volume of 56 billion. Now, MasterCard is integrating AI into its platform to serve its customers better. But in this process, MasterCard also encountered the following challenges:

- The deployment time is long, and a large number of deep learning modules need to be rebuilt on existing systems of MasterCard.
- The compatibility with other MasterCard IT modules is poor, such as inability to utilize existing ETL, data warehouses and other analysis-related data technologies and tool sets.
- Data needs to be copied frequently between different modules, and I/O performance becomes a bottleneck.

In order to address these challenges, MasterCard cooperated with Intel and introduced Analytics Zoo "Big Data Analytics + AI" platform to build a deep learning-based recommendation algorithm. Based on the latest research and industry practice, NCF and WAD models were selected as candidates for the recommendation model, and Keras-style API from Analytics Zoo is also used to build a deep learning model on Python and Scala.

After the model is built, by using Analytic Zoo's service APIs, MasterCard embedded deep learning and model service processes into the enterprise data pipeline built on Apache NiFi.

In order to validate the deep learning recommendation algorithm based on Analytics Zoo, MasterCard conducted benchmark tests on Spark Machine Learning and Analytics Zoo's BigDL model, in which the former test chose the Alternating Least Squares (ALS) model of Spark MLlib method. The comparison block diagram between the deep learning model and ALS model is shown in Figure 2-3-8.

Figure 2-3-8 Comparison between Deep Learning Model and ALS model

■ Solution Configuration and Results

In the comparison solution, data sets collected by MasterCard from specific channels in the past three years are used, including:

- Different Qualified Consumers: 675,000
- Target Merchants (offers or campaigns) for Benchmark: 2,000
- Known Transactions: 1.4 billion (53 GB raw data)
- Consumption Time: 12 - 24 months for training and 1-2 months for validation

The comparison between the deep learning model and ALS model of MasterCard recommender system is mainly based on the following four indicators:

1. Area Under the ROC Curve (ROC AUC)
2. Area Under the Precision and Recall Curve (PR AUC)
3. Accuracy and Recall Rate
4. Top 20% Accuracy of Each Customer

From the validation results, we can find that the deep learning model significantly improves over ALS model, as shown in the following table:

	NCF Model	WAD Model
Compared with ALS, the recall rate reduces by	29% ⬆	26% ⬆
Compared with ALS, the accuracy improves by	18% ⬆	21% ⬆
Compared with ALS, the top 20% accuracy improves by	14% ⬆	16% ⬆

Table 1 Better Results of Deep Learning Model over ALS Model

Conclusion

Finance, as a traditional industry focusing on data and processes, has accumulated a large amount of data in many years of operation. With the aid of AI applications, such data can be used to discover more values, support various businesses, provide more personalized services to end users, and improve user experience.

Using the end-to-end AI and big data analytics capabilities provided by Analytics Zoo, as well as a large number of models and APIs within Analytics Zoo, financial companies can quickly use their own data resources to build recommender systems based on NCF, WAD and other deep learning models on their existing big data platforms, such as Hadoop and Spark, without having to build from scratch, thereby greatly reducing the cost and time for financial companies to build business recommender systems.

In cases like China Life Insurance Shanghai Data Center and MasterCard, the solutions all adopted hardware platforms built on Intel® Xeon® processors/Intel® Xeon® Scalable Processor. In the future, users can also choose the next generation of hardware products such as the 2nd Generation Intel® Xeon® Scalable Processor with better performance and more optimizations for AI, thereby building solutions with better performance and stronger AI training/inference capability.

Accelerate AI Image Analysis, Empowering Insurance Industry

AI Accelerating Image Analysis for Insurance Industry

Image Analysis in Insurance Industry

Every insurance policy in the insurance industry has a huge demand for image analysis. For example, when buying car insurance or requesting for claim, the policyholder must upload certificates such as ID card, driving license and vehicle certificate to the insurance system for review by insurance company There are dozens of commonly used certificates and signatures, so manual review is not only time-consuming and laborious, but also prone to errors. In addition, taking the increasingly popular health insurance for example, the insurance company also needs to interpret the X-ray, CT and other images of the policyholder in order to accurately assess the policyholder's short-term and long-term health conditions.

Enhance AI Capability and Improve User Experience

At present, a variety of AI applications based on image analysis, including face detection & recognition and image segmentation, are being used more and more widely in the insurance industry. By embedding AI image analysis application into insurance business operation, risk management, intelligent customer service and internal control, the insurance company can effectively recognize risks and optimize business processes, enabling the insurance industry empowered by AI. For example, in the above-mentioned car insurance and health insurance cases, the insurance company can combine AI image analysis with NLP technology, to quickly find out necessary claim materials, automatically extract review information, then calculate claims according to settlement rules and risk control model, thereby automatically and efficiently completing the claims.

Intel has necessary algorithms and models for various modules such as face detection, comparison, recognition and liveness detection, to meet AI application requirements in this field. For example, OpenVINO™ tool kit released by Intel has provided dozens of pre-trained AI models, allowing users to immediately build AI applications such as face detection and recognition without starting from scratch.

* For more technical details about OpenVINO™ tool kit, please refer to relevant content in the Technologies section of this guide.

Let's take face liveness detection as an example. FeatherNet is a lightweight Convolutional Neural Network (CNN) developed by Intel and Huazhong University of Science and Technology for anti-fraud face recognition applications. Compared with traditional CNN, FeatherNet* has two main characteristics: firstly, Global Average Pooling (GAP) is replaced by Streaming Model. Although GAP can be used to reduce dimension and prevent over-fitting in many deep neural networks, but due to its lack of ability to distinguish regional weights, it is easy to reduce accuracy in face recognition scenarios. So FeatherNet has been added a streaming module with DWConv layer to replace GAP, which has greatly improved the accuracy. Secondly, FeatherNet constructs a new fusion classifier for multi-modal data fusion, which can combine and cascade the models learned from multi-modal data to improve the accuracy of model[13].

ResNet-based Deep Learning Technology

■ ResNet Introduction

Deep neural network is now one of the most widely used network models in AI image analysis. In the classical deep neural network, the more layers of the network, the richer the features of different levels can be extracted. At the same time, deeper networks can make the extracted features more abstract and richer in semantic information.

However, as the depth increases, the degradation problem also arises. In other words, the accuracy rate will first rise until reaching an upper limit, but as the depth continues to increase, the accuracy rate will gradually decline. The Residual Net (ResNet) can effectively solve this problem. As shown in Figure 2-4-1, several residual block structures can be built in ResNet, and their inputs and expected outputs are equal, thus establishing a constant mapping relationship.

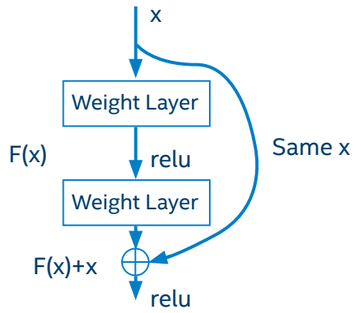


Figure 2-4-1 ResNet Residual Block Structure

With this structure, the deep neural network can keep the accuracy while increasing the depth. At present, ResNet has been widely used in AI application scenarios such as image recognition.

■ Model Implementation

Intel® Optimization for Caffe provides an optimized version of caffe prototxt file for RESNET50 network at:

1. CAFFE_HOME/models/intel_optimized_models/resnet50_v1/resnet50_fp32_acc.prototxt

The prototxt file using dummy data is located at:

1. CAFFE_HOME/models/intel_optimized_models/resnet50_v1/resnet50_fp32_perf.prototxt.

[13] For detailed description of FeatherNet technology and performance, please refer to https://arxiv.org/pdf/1904.09290.pdf.

■ Optimize code efficiency on Intel® Xeon® processor platform

■ Installation of Intel® Optimization for Caffe

The installation method of Intel® Optimization for Caffe 1.1.6 is as follows:

```
1. git clone -b 1.1.6 --recursive https://github.com/intel/caffe.git intelcaffe
2. cd intelcaffe
3. ./scripts/prepare_env.sh
4. mkdir build
5. cd build
6. cmake ..
7. build -j
```

Add the python directory of Intel® Optimization for Caffe to the environment variable Pythonpath:

```
1. export PYTHONPATH=/... /intelcaffe/python:$PYTHONPATH
```

■ Memory Optimization of Intel® Optimization for Caffe[14]

By default, Intel® Optimization for Caffe allocates a separate output buffer for each layer. Because the output buffer uses a different memory address, rather than being contained within the local memory cache, many memory lookups in layer forwarding will result in potential cache misses. Therefore, the circular buffer sharing mechanism, i.e. the method of reusing pre-allocated memory buffers across layers, can be used to reduce the cache miss ratio in Intel® Optimization for Caffe. In the compilation phase, the maximum size of the output buffer is determined by graph traversal. In the execution phase, once a layer completes execution, the layer's memory buffer will be released and put back into the circular queue for reuse.

At the same time, in multi-instance execution, weight sharing technology can also be used to provide better performance for the system. The mechanism of weight sharing is to improve the cache hit ratio between processor L3 Cache and memory by sharing weight buffers among multiple processes in the same NUMA node.

■ Utilize the Characteristics of NUMA to Control the Use of Processor Computing Resources

The data center usually introduces NUMA technology to run many servers as if they were a single system. A processor can access its own local memory faster than non-local memory. In order to obtain better computing performance in such system, it needs to be controlled by some specific instructions. **numactl** is a technical mechanism for controlling processes and shared storage. It is a widely used method for controlling computing resources in Linux. When running inference, Intel® Optimization for Caffe can also use numactl commands to improve the efficiency of computation and throughput.

The specific usage is as follows:

```
1. numactl -c 0-19,40-59 caffe time --forward_only --phase TEST -model MODEL_NAME
```

When execution, only the 0-19 and 40-59 cores in processor #CPU0 and the near-end memory corresponding to processor #CPU0 are used. Then correspondingly, a similar command can also run on the processor #CPU1:

```
1. numactl -c 20-39,60-79 caffe time --forward_only --phase TEST -model MODEL_NAME
```

In addition, due to the limitation on parallelism, the efficiency of many codes cannot greatly improve. If these tasks run on fewer processor cores, the efficiency will be higher. Therefore, if the processor core is bound by numactl to run more instances, higher throughput can often be achieved. Although the delay may increase, it is usually within the acceptable range of applications.

3D V-Net Segmentation Network-based Deep Learning Technology

V-Net uses an end-to-end trained fully convolutional network to process 3D image data and complete image segmentation. The network model no longer slices the data, but uses the 3D convolutional network layer to directly process 3D data. In addition, it can also use the objective function customized based on similarity coefficient to guide network training, thus optimizing training and improving speed.

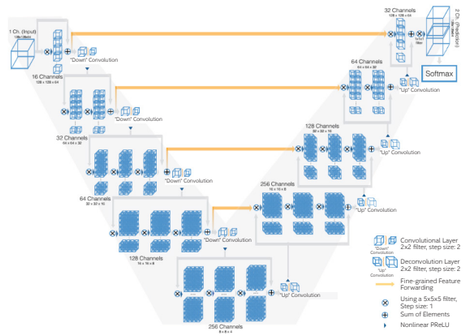


Figure 2-4-2 Graph of V-Net Convolutional Neural Network

The data size is limited to 128×128×64 3D data, and Batch Size=1 is set to estimate the memory load of the convolutional network. Bias should be taken into account when calculating the memory occupied by parameters, so the memory load required by the data is 313.7864 m × 4 bytes = 1255.1456 MB ≈ 1.23 GB (note that this is only a forward calculation process, and the backward calculation process needs twice as much memory as this)

- The memory load required by weight values is 77.44695 m × 4 bytes = 309.7878 MB ≈ 0.303 GB.
- The memory load required by the entire network model is about 1.23 + 0.303 ≈ 1.53 GB.
- If the memory is doubled for temporary storage, 3.06GB will be required.

[14] For detailed technical description, please refer to: https://arxiv.org/abs/1805.08691

Layer (batch-size=1)		Data (M)	Weight (M)
Input		1.048576	0
Conv1	Conv3D11	16.77722	0.002016
	Conv3D12	16.77722	0.032016
	Conv3D13	16.77722	0.032016
Pool1	Conv3D	4.194304	0.004128
Conv2	Conv3D21	4.194304	0.128032
	Conv3D22	4.194304	0.128032
	Conv3D23	4.194304	0.128032
Pool2	Conv3D	1.048576	0.016448
Conv3	Conv3D31	1.048576	0.512064
	Conv3D32	1.048576	0.512064
	Conv3D33	1.048576	0.512064
Pool3	Conv3D	0.262144	0.065664
Conv4	Conv3D41	0.262144	2.048128
	Conv3D42	0.262144	2.048128
	Conv3D43	0.262144	2.048128
Pool4	Conv3D	0.065536	0.2624
Bottom	Conv3D51	0.065536	8.192256
	Conv3D52	0.065536	8.192256
	Conv3D53	0.065536	8.192256
Deconv4	Deconv3D41	0.524288	0.524544
	rConv3D41	0.524288	8.192256
	rConv3D42	0.524288	8.192256
	rConv3D43	0.524288	8.192256
	rConv3D44	0.524288	8.192256
Deconv3	Deconv3D31	2.097152	0.262272
	rConv3D31	2.097152	2.048128
	rConv3D32	2.097152	2.048128
	rConv3D33	2.097152	2.048128
	rConv3D34	2.097152	2.048128
Deconv2	Deconv3D21	11.01005	0.0656
	rConv3D21	11.01005	0.512064
	rConv3D22	11.01005	0.512064
	rConv3D23	11.01005	0.512064
	rConv3D24	11.01005	0.512064
Deconv1	Deconv3D11	33.55443	0.016416
	rConv3D11	33.55443	0.128032
	rConv3D12	33.55443	0.128032
	rConv3D13	33.55443	0.128032
	rConv3D14	33.55443	0.128032
Softmax	Conv3D	2.097152	0.000066
	Output	2.097152	0
Total		313.7864	77.44695

Performance Gains from Intel® Architecture

■ AI Enhancement Features of Intel® Xeon® Scalable Processor

The next generation of Intel® Xeon® scalable processor with innovative microarchitecture have more cores, more concurrent threads and more cache. Meanwhile, it integrates a large number of hardware enhancement technologies, especially Intel® AVX-512 and other technologies, which can provide strong parallel computing capability for AI inference process and enable users to obtain better deep learning results.

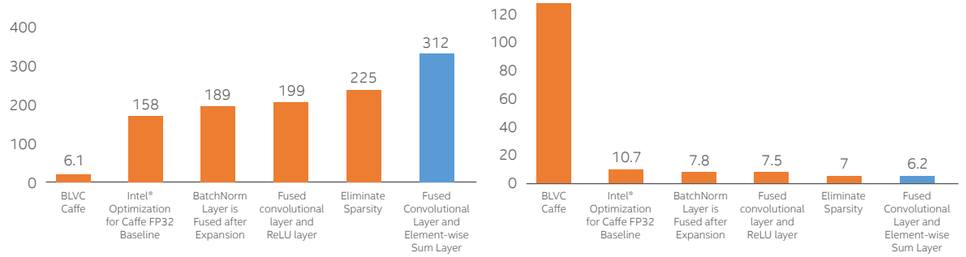


Figure 2-4-3 Intel® Optimization for Caffe Compared with BLVC Caffe in Inference Throughput and Inference Time Performance after being Optimized on Intel® Xeon® Scalable Processor

*For more technology details about Intel® Optimization for Caffe, please refer to the relevant description in the Technologies section of this manual.

Meanwhile, the Intel® processor has been optimized a lot for many popular AI frameworks, such as BVLC Caffe, TensorFlow and Apache MXNet. Take Intel® Optimization for Caffe as an example. Compared with BVLC Caffe, it further releases the advantages of Intel® Xeon® scalable processor[15] and achieves the effect of 1+1>2.

■ Methods and Codes of Intel® Optimization for Caffe

Intel® Optimization for Caffe invokes the optimized instruction set by invoking the API of Intel® MKL-DNN inside the layer, greatly improving the instruction parallelization of the program. Intel® MKL-DNN automatically invokes the Intel® AVX-512 instruction set built in the Intel® Xeon® Scalable Processor and the deep learning boost (VNNI instruction set) built in the 2nd Generation Intel® Xeon® Scalable Processor.

* For more technology details about the 2nd Generation Intel® Xeon® Scalable Processor and VNNI instruction set, please refer to the relevant description in the Technologies section of this manual.

Layer Fusion

Layer Fusion technologies, such as BN+Scale, Conv+Sum, Conv+Relu, BN InPlace and Sparse Fusion, can be used to improve the performance of deep learning. The combination of Layer Fusion and the Intel® Optimization for Caffe framework enables convolutional neural network such as ResNet to perform 2D image inference on Intel® Xeon® scalable processor platform with performance comparable to or even better than existing platforms. Meanwhile, they also provide good support for INT8 precision inference optimized by the VNNI instruction set, and tools such as calibration provided by the framework allow users to seamlessly switch the neural network to INT8, thus achieving greater performance improvement.

According to the statistics, when making a comparison between BVLC Caffe and Intel® Optimization for Caffe that adds Layer Fusion technology while running on Intel® Xeon® scalable processor, and uses ResNet50 convolutional neural network to perform AI inference in the same measurement environment, as shown in Figure 2-4-3, the inference performance per unit time increases by 51 times as much as that of the former, and the inference time is shortened to 4.7% of the former[16].

[15] Intel® Optimization for Caffe official website: https://github.com/intel/caffe

[16] This data is quoted from the article Highly Efficient 8-bit Low Precision Inference of Convolutional Neural Networks with Intel Caffe: https://arxiv.org/pdf/1805.08691.pdf; the test configuration is as follows: Convolution Model: ResNet50, Hardware: AWS single-socket c5.18xlarge.

Recommended Hardware and Software Configuration

For the construction of the above AI-based image analysis model, please refer to the following platforms based on Intel® architecture. The environment configuration is as follows:

Hardware Configuration

Name	Specification
Number of Nodes	1
Socket	2
Processor	Intel® Xeon® Gold 6148 processor or higher
Base Frequency	2.40Ghz
Cores/Threads	20/40
HT	On
Turbo	On
Memory	192G (16G DDR4 2666MHz x12)
BIOS	1.46

Software Configuration

Name	Specification
Operating System	Ubuntu 16.04
Linux Kernel	3.10.0-693.21.1.el7.x86_64
Workloads	Resnet50/VNet
Base Frequency	Gcc 4.8.5
Deep Learning Acceleration Library	Latest version of Intel® MKL-DNN
Deep Learning Framework	Release of Intel® Optimization for Caffe

Use Case: China Ping An

Background

Health insurance is one of the important types of commercial insurance. As people pay more attention to critical illness insurance, the premium scale, product types and coverage of this type of insurance are also gradually expanding. According to the statistics from China Banking and Insurance Regulatory Commission, by the end of 2018, the original insurance premium income of the health insurance business reached RMB 544.813 billion, up 24.12% year-on-year, accounting for 14.33% of the total original insurance premium income[17].

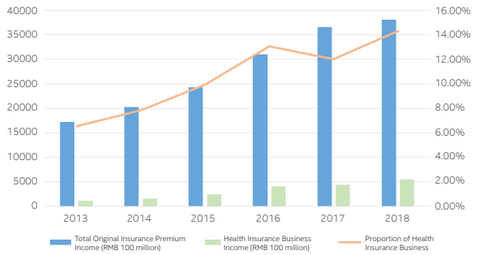


Figure 2-4-4 Trend of Proportion of Domestic Health Insurance Business in Recent Years

In comparison with the more sophisticated insurance market, there is still great room for improvement with regard to this figure. Take the United States as an example. Its commercial health insurance premium accounts for about 40% of the total premium[18], from which we can foresee the huge market potential of health insurance in China. However, in the process of rapid development, the health insurance market is also restricted by some weaknesses.

Compared with other types of insurance, the subject matter of health insurance is the health of the insured. Insurance companies need to assess the health status and accidental injuries of the insured accurately and clearly in order to reduce the operational risks of health insurance and control the loss ratio. However, the technical and management difficulty of this work is far more complicated than those of other types of insurance, and it requires relevant staff to have highly professional pathological knowledge and practical experience.

Medical imaging is not only the most frequently used diagnosis and treatment basis for medical institutions, but also an important basis for insurance institutions to determine the health status of the insured. Although medical imaging equipment has been widely used in hospitals at all levels, there are challenges facing the accurate interpretation of medical images. Image reading doctors need not only specialized knowledge in areas such as clinical medicine and medical imaging, but also relevant skills in radiology, CT, nuclear magnetic resonance, and ultrasonics. Moreover, they also need the ability to use various imaging diagnostic techniques to diagnose. Therefore, normally speaking, only experienced imaging doctors have the ability of accurate interpretation.

Now, the use of advanced AI technology to assist in medical image interpretation can not only effectively segment and locate the image, but also identify micropathological features that are difficult to find out with naked eyes through in-depth analysis of the image, thus improving the early detection probability of various malignant diseases. Therefore, AI-based 2D/3D medical image training and inference can not only effectively help major medical institutions improve their diagnosis and treatment efficiency, but also provide effective means for insurance institutions to conduct accurate health assessment.

Solution

In the concept of machine learning or deep learning, the AI model obtained by training is applied to new data, and this process is called inference. In medical image interpretation, the model obtained with training is used to infer and determine pathological features. Therefore, the inference efficiency is directly related to the efficiency of medical image interpretation.

In this case, Ping An is making AI inference on 2D/3D medical images using ResNET which has excellent performance in 2D image classification, detection and localization, the cutting-edge 3D image segmentation model V-Net segmentation network, and the deep learning framework such as Intel® Optimization for Caffe.

As shown in Figure 2-4-5, the inference process of Ping An medical image application is mainly divided into four stages: In the pre-processing stage, the system will perform operations such as medical image slice, ROI selection, data enhancement and normalization input preparation. In the inference engine, the system will use ResNet network and 3D V-Net segmentation network to run on the Intel® Optimization for Caffe framework, perform inference operation on the input medical images and obtain the processing results. In the post-processing stage, the system will carry out various morphological processing and merge the prediction results according to requirements. In the final stage of application layer processing, the system can output and display the results in XML and other ways.

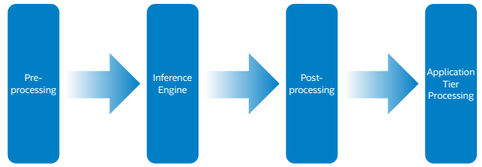


Figure 2-4-5 AI Inference Process of Ping An Medical Image

With ResNet network and 3D V-Net segmentation network, Ping An's AI team has efficiently carried out AI inference and achieved gratifying results. As shown in Figure 2-4-6, after using these leading technologies to segment the focus of optical coherence tomography (OCT) images of ophthalmic diseases such as age-related macular degeneration, it can be clearly seen with the results that the patient's intraretinal effusion (marked by yellow line), subretinal effusion (marked by red line) and subretinal hyper-reflective material (marked by purple line) can be clearly marked by intelligent application. The work that used to require experienced doctors to do with time and effort can now be done in seconds through AI applications.

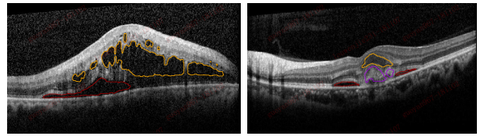


Figure 2-4-6 Results of OCT Lesion Segmentation Based on Intelligent Application

Through repeated laboratory and clinical training and inference, Ping An intelligent medical image analysis has achieved remarkable results in many application scenarios, for example, in the detection of lung nodules. The early identification of lung nodules is a good way to reduce the mortality rate of lung cancer. It can screen the high-risk population through low-dose CT scan so as to identify the risk as early as possible. However, the massive CT images resulting from this also greatly hinder the screening efficiency, while the use of AI for lung nodule detection can significantly improve the screening efficiency. In the Lung Nodule Analysis (LUNA) evaluation in early 2018, Ping An not only won the first place in the world by virtue of "Ping An Intelligent Lung Nodule Image Reading Technology", but also broke the world records of "lung nodule detection" and "false positive screening" with a precision of 95.1% and 96.8% respectively[19]. On the other hand, efficient medical image analysis can also accurately analyze whether the verified insurance is fraudulent, thus greatly improving the anti-fraud ability of Ping An insurance business.

Conclusion

AI-based image analysis can effectively help financial institutions improve their productivity, prevent fraud risks and improve user experience. Through deep learning frameworks such as Caffe and TensorFlow, such applications have been widely used in intelligent underwriting processes in the insurance industry for scenarios including pathological image interpretation, and bill processing.

The introduction of the Intel® Xeon® scalable processor and the deep learning framework optimized for Intel® architecture into these intelligent applications can not only effectively improve the inference efficiency of intelligent applications, but also enhance the landing capability and deployability of applications with higher cost performance and accelerate the application of AI in the insurance industry.

[17] This data is quoted from the official website of China Banking and Insurance Regulatory Commission: http://bxjg.circ.gov.cn/web/site0/tab5257/info4132154.htm

[18] This data is quoted from a media report: http://www.chinairn.com/news/20150211/140425992.shtml

[19] Data quoted from Luna official website: https://luna16.grand-challenge.org/Results/

Maintain Data Security and Break Down Data Silos to Provide Richer Data Sources for AI Applications

Explore the Use of Multi-source Data in AI Applications with the Help of Federated Learning Approach

AI and Federated Learning

■ Importance of dataset diversity for AI development

Thanks to the continuous development of algorithms, computing power and data, Artificial Intelligence (AI) technology has also made huge breakthroughs in the past decade, and gradually adopted in the financial, healthcare, manufacturing and other industries. During this process, the size and quality of the training dataset is profoundly influencing the AI performance. As shown in Figure 2-5-1, the research data indicate that the larger the size of the user training dataset, the better training results obtained[20].

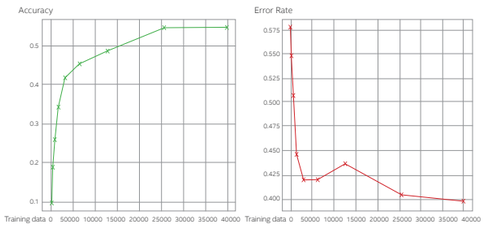


Figure 2-5-1 Larger training datasets lead to better training results

Meanwhile, some studies have also shown that larger training datasets can also effectively address the data imbalance problem commonly found in AI training in the financial industry[21]. Therefore, the larger and higher quality datasets, combined with increasingly sophisticated algorithms and greater computing power, will be an important factor in driving AI performance.

In practice, however, the massive datasets required for AI training are often distributed among data sources belonging to different enterprises and departments, and are isolated from each other for data security reasons. This so-called "data-silo" phenomenon will obviously affect AI training in the financial industry. Traditionally, when multiple financial enterprises or departments want to co-train a model, their data need to be integrated into one of them by using a distributed system, but this simple data integration cannot guarantee the security of data interactions, and greatly increases the risk of data privacy breach.

Data security and privacy are now a growing concern, and subject to more detailed and sophisticated laws and regulations. For example, the Administration Measures on Data Security (Draft for Comment)" released in May 2019 by the Cyberspace Administration of China in conjunction with relevant departments, made clear requirements on data processing and use as well as data security supervision and administration.

Nowadays, the contradiction between the desire for massive amount of high-quality training data from multiple sources and the concern for data security has undoubtedly become a great challenge that hinders the development and application of AI in various industries, especially in the financial industry. To effectively address this challenge, in 2016, researchers from Google AI proposed the federated learning approach, which is designed to train deep learning networks in environments where data silos exist. By using this federated learning approach, AI training can use data from different users, departments, and enterprises while ensuring privacy and security.

■ Mainstream federated learning method

The basic process of the current mainstream federated learning is shown in Figure 2-5-2. For example, in a scenario where two financial enterprises A and B implement collaborative training on a risk control model, the business systems of A and B each have a large number of credit card transaction data. For data security reasons, these highly sensitive data is stored in their respective data centers, and the firewall is deployed to enforce strict isolation, so that any direct access to these data will be denied.

To train the two data sources by using vertical federated learning, the first step is to align the data. Since data samples from different data sources are not identical and financial enterprises A and B will not disclose their respective data, the federated learning system needs to identify data samples common to both A and B (i.e., common users) by using encrypted data entity alignment method, as shown in the left half of Figure 2-5-2, in order to join the features of these data for modeling purposes.

When training these two data sources by using horizontal federated learning, we also need to begin with feature alignment. Since the feature dimensions of different data sources are not identical and financial enterprises A and B will not disclose their respective data, the federated learning system needs to identify feature dimensions common to both A and B (i.e., common features) by using encrypted data entity alignment method, as shown in the left half of Figure 2-5-2, in order to join the features of these data for modeling purposes.

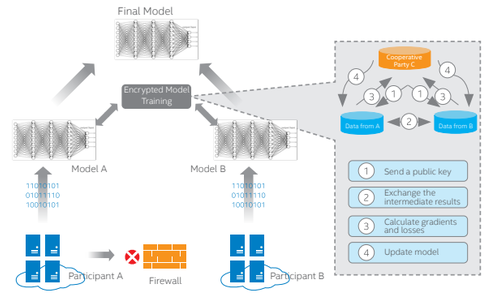


Figure 2-5-2 Basic Architecture of Federated Learning

[20] Data and graphs cited from De Berker, A., Predicting the Performance of Deep Learning Models, https://medium.com/@archydeberker/predicting-the-performance-of-deep-learning-models-9cb50cf0b62a

[21] Conclusions cited from Juba, B. and H. S. Le, Precision-Recall Versus Accuracy and the Role of Large Data Sets, Association for the Advancement of Artificial Intelligence, 2018.

Once the common data samples and common features are identified, both A and B can use these samples for model training. In order to ensure data confidentiality during the training process, as shown in the right half of Figure 2-5-2, it is necessary to perform encrypted training with the help of a cooperative party C. The encrypted training process is divided into the following four steps:

Step 1: The cooperative party C distributes the public key to A and B to encrypt the data to be exchanged in the training process.

Step 2: A and B interact with each other to calculate the intermediate result of gradient in encrypted form.

Step 3: A and B perform calculations with the encrypted gradient values, respectively, and provide the results to the cooperative party C. The cooperative party C calculates and decrypts the total gradient value by aggregating the results.

Step 4: The cooperative party C sends the decrypted gradient back to A and B, respectively, and A and B update the parameters of their respective models with the decrypted gradient.

The above training steps will iterate until the loss function converges, the training process is complete and the final model is obtained. It can be seen that, compared to a general distributed machine learning/deep learning approach, the federated learning approach has the following characteristics:

- Data is localized: participants train the global model using data they own.
- Every participant is involved in the learning process and model losses are controllable.
- The training process strikes a balance between privacy and security, and participants are able to co-construct the model without disclosing the underlying data and its encrypted form.

In addition, the federated learning also features an effect-based incentive mechanism, i.e., after the model is built, the effectiveness of the model can be evaluated in actual applications and documented by a permanent data logging mechanism (e.g. blockchain). Participants who provide more data get better model results, and model results depend on the contributions that data providers make to themselves and others. The effects of these models are delivered to data sources in the federated mechanism and continue to encourage other data sources to participate in the federated learning.

Based on the above characteristics, the federated learning can provide a cross-enterprise and cross-department data usage and model building method for AI applications in various industries, keep the private data from each data source localized, and build a learning model optimization mechanism without violating data privacy regulations by exchanging parameters in encrypted way.

For source code of the Google federated learning, please refer to:
https://www.tensorflow.org/federated/

As described in the preceding example, there is more than one way to perform federated learning. Depending on the dataset, the federated learning can be divided into Horizontal Federated Learning, Vertical Federated Learning, and Federated Transfer Learning. Specifically, the Horizontal Federated Learning applies to scenarios where there are a lot of overlaps on user features but only a few on users between different datasets. It can slice the dataset by user dimension and retrieve data with the same feature but different user for training. For example, in the same commercial bank, user data from different branches can be trained in a horizontal federated learning manner.

The Vertical Federated Learning, on the other hand, applies to scenarios where there are a lot of overlaps on users but only a few on user features between different datasets. This method allows you to slice the dataset by feature dimension and retrieve data with the same user and different feature for training. For example, in a financial group, data from the same group of users in the insurance and credit card businesses can be trained in a vertical federated learning manner. In scenarios where there are a few overlaps on both users and user features, the Federated Transfer Learning can be used to perform collaborative data training in a transfer learning manner, instead of slicing the data.

■ Application of federated learning solution in the financial industry

Ever since it appeared, the federated learning has received tremendous attention, and has evolved into many industry-specific solutions amid the continuous efforts by users from various industries. Although the financial industry has always attached great importance to the IT technologies and has accumulated rich business data in its long-term operations, the large-scale conglomerates commonly found in the financial industry and the highly sensitive nature of financial data have created data silos between different departments, branches, and enterprises.

As AI applications such as intelligent risk control, targeted marketing and anti-fraud are important engines that drive the business transformation of financial enterprises, the federated learning is increasingly becoming a powerful tool for the financial industry to effectively maintain data security, break down data silos and provide richer data sources for AI applications.

According to the federated learning process in the previous section, when financial enterprises perform the AI model training with data from multiple sources by using the federated learning approach, the AI model, data or process parameters need to be transmitted and interacted across data nodes on the network. It is well known that the greater the exposure of data, the greater the security risk it faces. Thus, both the vulnerabilities in each node (for example in hardware or operating system) and the compromised network devices such as routers and gateways will expose organizations to the risk of a data breach or tampering.

For example, a hacker may intercept data packets by installing a sniffer on the network forwarding device, or may exploit the cold boot attack to dump the residual data after server reboot, or may directly attack the data in memory through memory bus snooping, memory tampering and other attacks. The system is not able to effectively protect against all these methods of attack. Building a bottom-up security strategy that covers software, hardware and operating system will not only consume huge resources and increase the total cost of ownership (TCO) of users, but also may not be as effective as desired.

Therefore, when building a federated learning system, the key is to create an efficient and trusted way for users to share data. Currently, solutions based on the Trusted Execution Environment (TEE) are becoming increasingly popular in the financial industry. The core concept behind these solutions is to provide a secure and trusted environment built on third-party hardware for different data sources. Figure 2-5-3 shows a recommended federated learning architecture that adopts the Trusted Execution Environment (TEE). As shown in the upper part of the figure, data from data sources A and B can be shared and trained in a TEE environment built on the hardware.

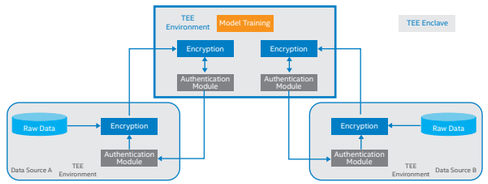


Figure 2-5-3 TEE Solution Architecture

As a typical implementation of TEE solution, Intel® Software Guard Extensions (Intel® SGX) is designed to create a trusted "enclave" within the specific hardware (for example in memory), as shown in Figure 2-5-4, in order to limit the security boundaries of data and applications to the enclave as well as the processor, while also making its operation independent of other software and hardware. This means that data is secured independent of the software, operating system or hardware configuration, which can prevent a data breach even if the hardware driver, the virtual machine or the operating system is compromised by an attack. In addition, as a hardware-based security technology, the SGX/TEE solution is unmatched by other technologies in terms of efficiency, and its advantages in affordability and usability are the most important factors that are valued by companies when building a federated learning system.

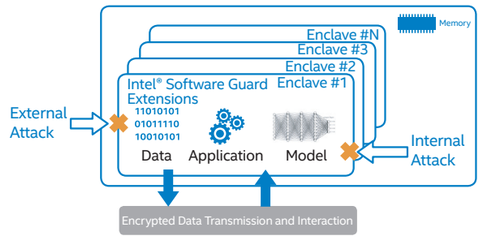


Figure 2-5-4 Intel® SGX Secures Data with the Trusted Enclave

Intel is now taking the lead in working with a number of financial partners to collaboratively implement AI training using multi-source data in a secure and trusted environment. This exploration has now achieved fruitful results and has received good feedback from customers in numerous projects.

Intel® SGX

■ Introduction to Intel® SGX

Intel® SGX, introduced by Intel in 2013, is a new set of instruction set extensions and access controls that enable isolated operation of different applications to enhance the security of application code and data, providing them with better protection against data breach or tampering.

Traditionally, protection for data privacy and security is mostly implemented at the operating system or software level, but when the operating system or software is compromised, the security of the data is in jeopardy. As shown in Figure 2-5-5, although applications can be protected against attacks from external hackers or malicious applications with the aid of security scans, firewalls, and other means, but malware and malicious code that exploit operating system vulnerabilities can bypass these protections and directly attack critical private data.

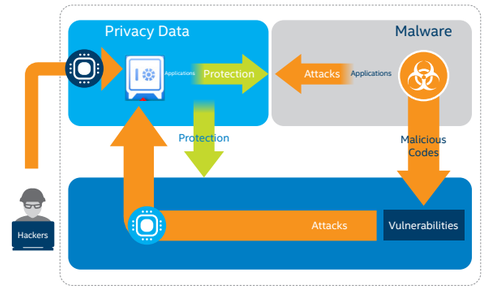


Figure 2-5-5 Internal Attacks on Applications

By the virtue of Intel® SGX, developers are allowed to store sensitive information or applications in the enclave. The enclave is an execution area created in a specific hardware (for example in memory), which provides stronger security protections, is independent of the security state of firmware and software, and has hardware-based confidentiality and integrity, thus enabling the system to deny access from higher privileged processes.

As a result, Intel® SGX can provide users with the following key features:

a) Enhanced confidentiality and integrity
As shown in Figure 2-5-6, the enclave works in an isolated hardware environment (Intel architecture processors and memory with SGX support) and authenticates applications and data with keys, thereby protecting data from attacks even if there is privileged malware or malicious code in the operating system (OS), BIOS, or virtual machine (VM).

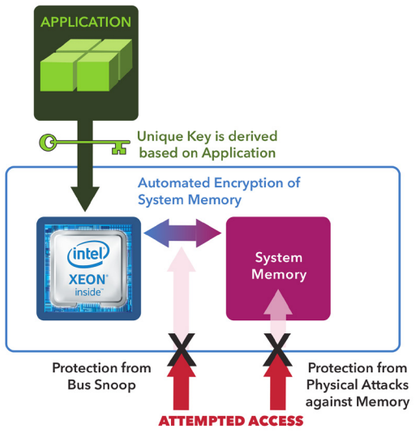


Figure 2-5-6 SGX Provides Enhanced Confidentiality and Integrity

b) Smaller security attack surface

As illustrated in Figure 2-5-7, the SGX technology confines applications and sensitive data in protected hardware enclaves, which eliminates traditional attacks that malicious programs can launch from hardware, virtual machines and operating systems, resulting in a smaller attack surface for greater security.

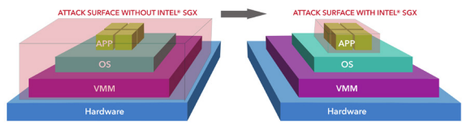


Figure 2-5-7 SGX Has a Smaller Security Attack Surface

c) Remote authentication and control capabilities

Users can verify that the operating environment is loaded in a legitimate enclave by performing remote authentication; thus, keys, credentials, and other sensitive data can be more securely provided to the enclave;

d) Lower learning curve

Applications using the SGX technology can be developed, integrated and executed on Intel® processors, eliminating the need for developers to be familiar with additional hardware and software environments and resulting in a lower learning curve.

■ Intel® SGX Installation and Configuration

Users can create an Intel® SGX-based solution by introducing Intel® SGX SDK which provides:

- API
- Function library
- Document
- Sample source code
- tools

The latest Intel® SGX SDK can be obtained by clicking the following link:
Download link of SDK for Windows
https://registrationcenter.intel.com/en/forms/?productid=2614
Download link of SDK for Linux
https://01.org/intel-software-guard-extensions/downloads

To use Intel® SGX, users need to first confirm that their current CPU supports the technology and that SGX is set to Enabled in the BIOS menu. The user can then refer to the installation guide that came with the SDK to complete the SDK installation process.

On hardware platforms that do not support Intel® SGX, users can also install the SDK and develop their SGX applications in the simulation mode, which behaves essentially the same as running in a real SGX-enabled hardware environment, but without the actual protection provided by SGX.

■ Intel® SGX Attestation

When using Intel® SGX, the attestation is a critical step in protecting the privacy and security of data, which provides systems with:

- The ID of applications or data in the enclave.
- Details about an unmeasured state (such as the execution mode).
- The possible application or data tampering in the enclave.

Intel® SGX currently offers two types of remote attestation: Intel® EPID (Intel® Enhanced Privacy ID) and Intel® SGX DCAP (Intel® Software Guard Extensions Data Center Attestation Primitives). The following is a brief description of each type of remote attestation.

Remote Attestation Based on Intel® EPID

The remote attestation based on Intel® EPID will attest applications or data in the enclave by using the Intel® EPID signature, which can minimize the complexity of handling multiple security versions for a platform with an Intel® SGX Trusted Computing Base (TCB).

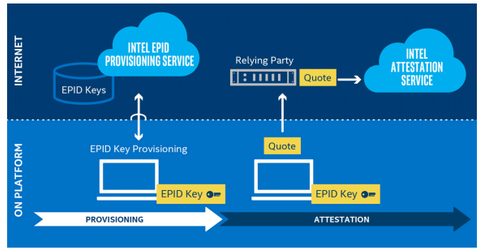


Figure 2-5-8 Remote Attestation Based on Intel® EPID

As shown in Figure 2-5-8, this type of attestation provides the device with an authorized EPID key, and in case that the private key used by the device is stolen or compromised, the EPID system automatically detect this situation and revokes the device to prevent the information from being compromised.

For more details on Intel® EPID security technology, see:
https://software.intel.com/content/www/us/en/develop/articles/intel-enhanced-privacy-id-epid-security-technology.html

For related code sample, see:
https://software.intel.com/content/www/us/en/develop/articles/code-sample-intel-software-guard-extensions-remote-attestation-end-to-end-example.html

Remote Attestation Based on Intel® SGX DCAP

The remote attestation based on Intel® SGX DCAP allows users to build and deploy their own attestation service, which is useful for enterprise, data center, and cloud service providers who need to address any of the following requirements:

- The network environment where the AI service is located does not have access to Internet services.
- The attestation needs to be strictly performed in-house.
- The AI service is deployed in a special network or architecture, such as a peer-to-peer network.

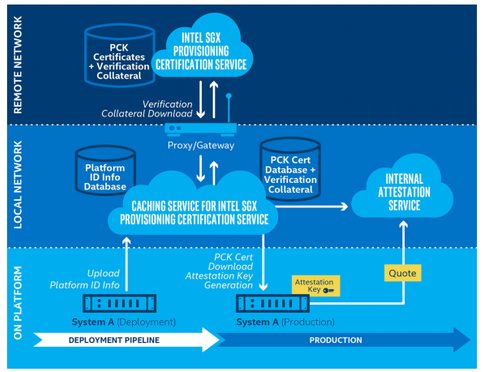


Figure 2-5-9 Remote Attestation Based on Intel® SGX DCAP

As shown in Figure 2-5-9, compared to the remote attestation based on Intel® EPID, the complete DCAP attestation process is performed in the user's own network environment, and the external access to attestation information is required only during the server deployment phase and the TCB Recovery phase. As a result, the DCAP attestation has lower latency and more flexible adaptability to the network environment in a data center environment.

Users can refer to the following technical documentation for more details:

- Data Center Attestation Orientation Guide: https://download.01.org/intel-sgx/dcap-1.1/linux/docs/Intel_SGX_DCAP_ECDSA_Orientation.pdf
- Installer and documentation for Linux: https://01.org/intel-software-guard-extensions/downloads
- Supporting Third Party Attestation for Intel® SGX DCAP: https://software.intel.com/sites/default/files/managed/f1/b8/intel-sgx-support-for-third-party-attestation.pdf
- Intel® SGX DCAP Source Code (including samples): https://github.com/intel/SGXDataCenterAttestationPrimitives

■ Developing and Porting Intel® SGX-based Application

When developing SGX-based applications, users can use the Trusted Libraries provided by the SGX SDK. The Trusted Libraries include most of the C/C++ APIs, as well as protected file systems, common cryptographic operations, multi-threaded support, and more. For all the APIs provided by the SGX SDK, see the SGX Developer Reference.

SGX Developer Reference for Windows
https://software.intel.com/content/www/us/en/develop/download/sgx-sdk-developer-reference-windows.html
SGX Developer Reference for Linux
https://01.org/intel-softwareguard-extensions/documentation/intelr-software-guard-extensions-documentation

To facilitate the development of AI applications, the SGX SDK specifically provides Intel® DNNL (Deep Neural Network Library) for deep neural networks, enabling users to invoke DNNL's standard APIs in the SGX trusted execution environment. Users can also easily port existing deep neural network applications developed based on the DNNL APIs and run them in the SGX enclave.

SGX DNNL Open Source Code:
https://github.com/intel/linux-sgx/tree/master/external/dnnl

When developing SGX-based AI applications, users typically protect the confidentiality of data in the following way:

- Node 1, which is responsible for computing, generates a secure random number in the SGX enclave as a key. The code running in Node 1 cannot pass the key, the data decrypted with the key and the intermediate results of the computation outside the enclave.

- Node 1 uses a remote attestation method to generate a report, certifying that its hardware and software environment is secure. Node 1 then sends a data request to Node 2 that provides data or gradients.
- Node 2 verifies the report of Node 1 to confirm whether Node 1 is secure. At the same time, Node 2 uses other methods to confirm whether Node 1 has access to the data.
- Node 2 encrypts the data with the key provided by Node 1 and sends it for decryption and use by Node 1.

When designing a federated learning application by using the above procedure, the following workflow can be implemented in SGX:
1. The application creates an instance of an SGX enclave and runs its own trusted code in it.
2. The trusted code of the application randomly generates an asymmetric key pair, and keeps the private key in the SGX enclave.
3. The application uses the remote attestation to generate a verifiable report of the hardware and software environment containing the public key generated in the above step 2 as the identity of the current node.
4. The application sends the remote attestation report and the public key to other nodes in the network. The other nodes record the public key of the current node as one of the trusted participants after verifying the report of the current node.
5. Repeat the steps 1-4 until all nodes have recorded the public keys of the other nodes.
6. Complete the encrypted data alignment and gradient exchange between nodes by using the federated learning algorithm. Each node uses the public keys of the other nodes to encrypt the data in subsequent communications.

■ Intel® SGX Integration with Graphene

Intel® SGX provides users with a hardware-based security mechanism that makes AI training and inference on critical data more secure and reliable. However, in actual deployments, users may face a new problem. For new AI applications, they can be integrated with the trusted libraries provided by the SGX SDK at the beginning of the coding, as described in the previous section. However, during the development of some AI applications, users may reuse some existing open source frameworks or algorithm implementations, such as neural network models based on Python and TensorFlow, which requires a lot of porting work.

Figure 2-5-10 Application Integration with Intel® SGX

As shown in Figure 2-5-10, users first need to port the application source code (written in C++, Python, or other languages) to support Intel® SGX, then compile and integrate it with the trusted libraries provided by Intel® SGX SDK to build the application, and finally execute it in a hardware trusted environment. This process requires not only Intel SGX porting expertise and experience, but also the deep understanding of the source code of the application being ported (e.g., open source frameworks, algorithm implementations). In some financial enterprises, the open source frameworks or algorithm implementations of their applications have been used for a long time, so the amount of work involved in modifying and porting source code may be enormous.

To reduce the burden of porting the above applications and eliminate the need for rewriting these codes in C/C++, users can choose to run these existing codes using a Library OS that supports Intel® SGX, such as Graphene (**https://github.com/oscarlab/graphene**), Occlum, Scone and Anjuna. It is important to note that if users choose the Library OS approach to address this challenge, they also need to properly configure the security options in Library OS and evaluate the existing codes loaded. Otherwise, incorrect software behavior may affect the performance, security, or stability of the application.

As an important open source compatibility tool for Intel® SGX, Graphene enables users to run their original applications directly in the Graphene SGX environment thanks to its support for dynamic loading libraries, dynamic linking, multi-process abstraction, and file authentication. Taking the TensorFlow framework as an example, as shown in Figure 2-5-11, the Graphene SGX environment can be built on the TEE environment in hardware, on which the user can create a Python runtime environment that runs in the Graphene SGX environment, and then run TensorFlow and the neural network model codes that the user needs in this Python runtime environment. Likewise, these applications can run in environments such as C++, where the OpenVINO™ toolkit and the Analytic Zoo platform which are commonly used in deep learning methods can also easily run in such environments.

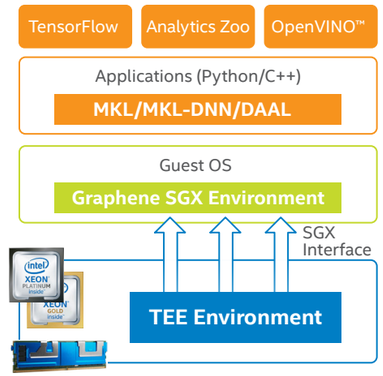


Figure 2-5-11 Running Applications in the Graphene SGX Environment

This process allows users to perform the required computing in the SGX environment without modifying the corresponding module code. Instead, Graphene performs encryption and semantic checks on untrusted host interfaces, and developers simply provide a manifest file to configure the application environment and isolation policy, and Graphene does the rest automatically.

■ Processors with Intel® SGX Support

It is worth noting that although AI applications built on cloud services (including public, private and hybrid clouds) become more popular in the financial industry, existing cloud server hardware may not all support Intel® SGX, which may affect the adoption of SGX technology in the financial enterprises. Those enterprises will either need to be patient and wait for the next hardware refresh cycle, or they will need to pay additional procurement costs. To address this challenge, Intel is gradually integrating the SGX technology into its processors, providing users with the ability to quickly deploy such technology.

Intel® Xeon® E-2200 processor is one of Intel's processor platforms designed to facilitate enterprises to build efficient and reliable IT systems, including cloud services, that drive sustainable business growth. This processor series includes 12 variants with SKU designation E-22xxG, and the number of cores/threads varies from 4/4 to 8/16. In terms of top processing speeds, the two high-end SKUs each have a maximum Turbo frequency of 5.0 GHz. Meanwhile, Intel® Xeon® E-2200 processor integrates Intel® SGX to provide the hardware-enhanced security.

For users, the intelligent 2nd Gen Intel® Xeon® E-series processors offer four key benefits:

- **Performance:** The 2nd Gen Intel® Xeon® E-series processors deliver up to 2X the performance of an entry-level server released in 2015, according to an Intel test.
- **Scalability:** The maximum memory on the 2nd Gen Intel® Xeon® E-series processors has doubled to 128GB, and the maximum memory bandwidth has increased by about 20% to 41.6 GB/sec.
- **Reliability:** The server management capabilities are built into the 2nd Gen Intel® Xeon® E-series processors with Intel® Active Management Technology (Intel® AMT), and the Intel Server Platform Services & Node Manager can be provided via a new firmware update.
- **Security:** Thanks to Intel® SGX, the hardware-enhanced security is an important part of the new processors.

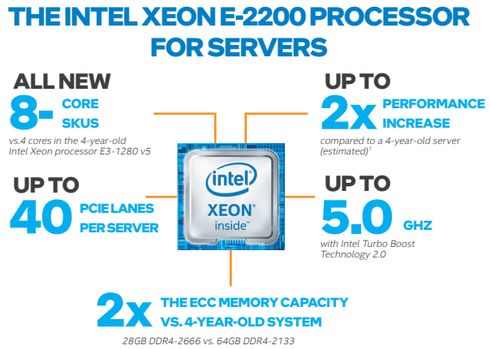


Figure 2-5-12 Intel® Xeon® E-2200 Processor

In addition, Intel is also planning to incorporate the Intel® SGX technology into more processor platforms, such as the Xeon series, allowing users to implement the hardware-enhanced security more easily and facilitating the adoption and development of AI methods such as federated learning.

Intel® SGX-based Solution

■ 1+N AI model training solution for multi-source data

With Intel® SGX technology, the financial enterprises can build a variety of solutions tailored to their needs. The following section briefly describes a 1+N AI model training solution for multi-source data built on Intel® SGX technology.

The 1+N solution architecture, shown in Figure 2-5-13, adopts a network that is composed of an aggregator enclave at the center and an edge enclave deployed in each data source. The aggregator enclave and the edge enclave in each data source are the trusted regions that require privileged access, and are constructed in memory by processor instructions provided by Intel® SGX.

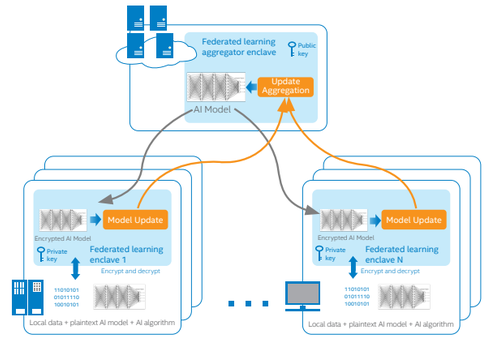


Figure 2-5-13 Federated Learning Solution with Intel® SGX

In the solution, the AI model to be trained and optimized and the related intermediate parameters are transmitted in the encrypted channel, while the training data, the plain-text AI model, and the AI algorithm are localized on each node. During the initialization process, each enclave generates a public-private key pair, and the public key is registered with the aggregator, while the private key is stored in the respective edge enclave. When the training begins, the aggregator first establishes an encrypted connection with a destination enclave. During this process, the asymmetric algorithm of the public-private key pair is used to negotiate a symmetric encryption key for this connection in order to prevent the man-in-the-middle attacks. After the connection is established, the aggregator will push the model that needs to be trained to each enclave in the encrypted form, and then each enclave will decrypt the model and send it to the local AI training environment to train the local data. At the end of the training, the local AI training environment returns the intermediate parameters obtained from the training to the local enclave.

The solution can also incorporate a number of innovative processes for business needs. For example, each enclave in the local environment is a federated trusted agent, and as later algorithms can run directly in the enclave, this trusted agent can perform more and more tasks in the local environment. Next, the enclave will return the intermediate parameters in the encrypted connection back to the aggregator enclave. The aggregator enclave will quickly aggregate the received intermediate parameters, and optimize and adjust the AI model based on the results before proceeding to the next round of iterations.

Since the above processes are implemented in enclaves, i.e., during the whole iterative cycle of the solution, the AI model and the intermediate parameters are passed through the encrypted channel and interacted in the encrypted enclaves, without any contact with external software and hardware, thereby establishing a secure and reliable internal loop. The Intel® SGX architecture-based processors provide powerful computing performance to build enclaves, establish encrypted channels, and interact and aggregate intermediate parameters.

The solution also gives a satisfactory practical approach for evaluating the contribution of each node to the training effect. In case there are N data sources in the solution, all nodes will participate in the training first to obtain an overall training effect. Then, N-1 nodes except the node to be evaluated will participate in the training (e.g., when evaluating Node #1, Nodes #2 to #N will participate in the training). After obtaining models with different training effects, the system can calculate the "contribution coefficient" of each data node in the federated learning, which in turn gives an accurate evaluation of the contribution of each data node in AI collaborative training, and adjusts the solution accordingly.

■ Building a Graphene-based deep learning training model

The Graphene tool allows users to build a deep learning training model quickly and easily in a hardware environment that supports Intel® SGX technology, and the following section briefly describes how to quickly build a Graphene SGX environment and run the TensorFlow framework on it for model training.

Taking the Ubuntu 18.04 and Python 3.6 environment as an example, the basic process is as follows:

a) Install Intel® SGX SDK, SGX PSW and SGX Driver

Download link: https://download.01.org/intel-sgx/ linux-<version>/<OS><OS version>

For details on installation, please refer to the following link: https://github.com/intel/linux-sgx/blob/master/README.md.

b) Install the Graphene tool:

Download the latest Graphene tool from Github:

```
git clone https://github.com/oscarlab/graphene.git
```

Run the Graphene-sgx environment by using the following steps (which can be adjusted according to the actual situation):

1. Install the SGX-related dependencies

```
sudo apt-get install -y libprotobuf-c-dev protobuf-c-compiler \
   libcurl4-openssl-dev

sudo apt-get install -y python3-protobuf
```

2. Installing a Linux kernel patched with FSGSBASE

```
git clone --single-branch --branch linux-5.4.y \
    https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git
cd linux
git am <graphene-dir>/Pal/src/host/Linux-SGX/sgx-driver/fsgsbase_patches/*.patch
```

Build and compile the kernel by following the instructions in the Ubuntu Wiki (**https://wiki.ubuntu.com/KernelTeam/GitKernelBuild**).

```
uname -r
LD_SHOW_AUXV=1 /bin/true | grep AT_HWCAP2
```

3. Generate the signature key

```
openssl genrsa -3 -out enclave-key.pem 3072
```

4. Compile

```
make SGX=1
```

For more detailed step-by-step instructions, please refer to the link: https://graphene.readthedocs.io/en/latest/building.html

After the installation is complete, if you can successfully run the Examples/python-scipy-insecure sample, then the installation is successful. Specific steps to run the sample can be found at the following link: https://github.com/oscarlab/graphene/tree/master/Examples/python-scipy-insecure

c) Download the **python.manifest.template** from the following link

https://raw.githubusercontent.com/oscarlab/graphene/master/Examples/python-scipy-insecure/python.manifest.template

Download to the **graphene/Examples/python-test** directory and run the following command:

```
$  python3.6 -m venv venv
$  ./venv/bin/pip install --upgrade pip
$  ./venv/bin/pip install --upgrade setuptools
$  ./venv/bin/pip install --upgrade wheel
$  ./venv/bin/pip install numpy==1.16.5
$  ./venv/bin/pip install tensorflow==1.14
$  sudo chown -R root:root venv
```

Modify the **python-test/python.manifest.template** file accordingly by adding the following:

```
loader.exec = file:venv/bin/python3.6
loader.argv0_override = venv/bin/python3.6
…
sgx.enclave_size = 4G
sgx.thread_num = 64
sgx.file_check_policy = allow_all_but_log
…
sgx.allowed_files.keras = file:keras
sgx.allowed_files.venv = file:venv
```

d) Prepare the **test_resnet.py** file to be tested:

```
import ctypes.util
def fl(name):
    raise PermissionError("No permission")
ctypes.util.find_library = fl

import os
import posix
import platform

def static_os_uname():
    return posix.uname_result(('Linux', 'sgx256', '4.15.0-20-generic', '#21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018', 'x86_64'))

def static_platform_uname():
    return platform.uname_result('Linux', 'sgx256', '4.15.0-20-generic', '#21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018', 'x86_64', 'x86_64')

os.uname = static_os_uname
platform.uname = static_platform_uname

import time, traceback
import numpy as np
import tensorflow as tf
import tensorflow.keras as k
import h5py, io

def main():
    model = tf.keras.applications.resnet50.ResNet50(weights=None)

    x_test = np.zeros((400, 224, 224, 3))

    for _ in range(0,10):
        start_time = time.time()
        model.predict(x_test, batch_size=4, verbose=1)
        print("Evaluation took", time.time() - start_time, "seconds")

main()
```

In this case, the dataset uses the **cifar10** open source dataset to construct the **Resnet50** network for training. The trained models are stored in **python-test/scripts/saved_models**.

Add the following statement to the program will allow the model to be saved properly.

```
os.environ["HDF5_USE_FILE_LOCKING"] = 'FALSE'
```

e) Under the **python-test** directory,

```
$  make SGX=1
$  SGX=1 ./pal_loader python3.6.manifest.sgx ./scripts/test_resnet.py
```

can run directly.

Recommended Hardware and Software Configuration

The above 1+N AI model training solution for multi-source data can be built on the following Intel® architecture-based platform with the following environment configuration:

Hardware Configuration

Name	Specification
Processor	Intel® Xeon® E-2200 processor or higher
Base Frequency	Maximum Turbo frequency of 5.0 GHz
Cores/Threads	8/16
HT	Off
Turbo	On
Memory	Up to 128GB DDR4 2666MHz
Hard Drive	Intel® DC S3320 SSD 480GB

Software Configuration

Name	Specification
Operating System	Ubuntu Linux release 18.04
Linux Kernel	4.15.0-74-generic
Workloads	LSTM, XGBoost
Compiler	GCC 7.4
Open Source Software Library	Graphene 1.0, Python 3.6.8, TensorFlow 1.14, XGBoost 0.9
SGX SDK	Intel SGX Linux 2.8 Release

Technology Outlook

The TEE-based federated learning architecture is an emerging technology that offers a new option for other federated learning implementations, and in this guide, we discussed the value of implementing TEE components, represented by Intel® SGX, in federated learning. Intel is currently working with a number of partners to promote the adoption of TEE in the federated learning architecture design, with an aim to build a feasible solution that is complementary with other implementations.

In the future, Intel and our partners also plan to further reinforce the TEE-based federated learning architecture discussed in this guide in the following ways:

1. Integrate this architecture into other Intel® Xeon® servers with SGX functionality, in accordance with Intel's product roadmap.
2. Optimize our algorithm to enable the loading of more federated learning algorithms into the SGX enclave.
3. Support the autonomous negotiation and distribution of federated learning key by using the blockchain.

Use Case: China Ping An

Background

As a technology solution expert under Ping An Group, Ping An Technology is committed to providing intelligent end-to-end technology services to facilitate the intelligent transformation of various enterprises. With their AI and cloud computing solutions, Ping An Technology have served more than 500 million users, covering five ecosystems of finance, healthcare, automobile, real estate and smart city.

In this regard, Ping An Technology hopes to aggregate multiple data sources to provide more and better training data sets for AI applications, thereby improving the training performance of the AI platform in different scenarios. However, the security risks associated with data access, aggregation, and interaction make it impossible to adequately protect the privacy of user data. That's why Ping An Technology urgently requires finding a better solution to fully exploit the value of data while ensuring data privacy.

Led by its Federated Learning Technology Team ("Federated Learning Team") consisting of a group of AI implementation experts, Ping An Technology has enhanced its AI model training performance by exploring and applying the federated learning method and aggregating more data. During this process, the Federated Learning Team has also engaged in a series of in-depth collaborations with Intel, and used Intel® SGX to successfully build a one-stop solution for AI training of multi-source data while protecting the data privacy and security, in other words the Hive Federated Learning Platform. The solution has achieved satisfactory results in several scenarios and provides a useful example for exploring the collaborative AI training of multi-source data.

Hive Federated Learning Platform from China Ping An

By using the Intel® SGX technology, the Hive Federated Learning Platform provides a one-stop solution for AI training of multi-source data while protecting the data privacy and security. The Platform features the following:

- Ability to provide a variety of encryption methods and support the homomorphic encryption and other secure multiparty computation mechanisms.
- Support for single-node and multi-node training;
- Ability to perform training by directly using an Intel architecture processor.
- Support for multiple deep learning/machine learning algorithms and frameworks.

The platform architecture is shown in Figure 2-5-14. The Hive Federated Learning Platform consists of four layers from bottom to top, in which the bottom layer is the infrastructure layer composed of a series of hardware devices, including processors based on the Intel® architecture; and the next upper layer is the operator layer, where the platform provides support for a variety of deep learning frameworks, including TensorFlow, Keras, PyTorch, and MXNet. In addition, the operator layer integrates several key federated learning functional modules such as sample alignment, feature alignment and gradient computing. The user's AI model, data, and parameters will invoke Intel® SGX via the modules within the operator layer, thereby building enclaves within the infrastructure layer to create a trusted AI training environment.

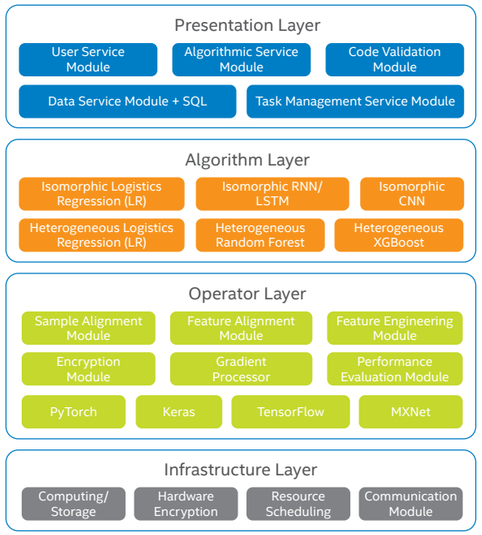


Figure 2-5-14 Hive Federated Learning Platform Architecture

In the algorithm layer, the Hive Federated Learning Platform integrates a large number of common deep learning/machine learning algorithms such as homogeneous logistics regression, homogeneous RNN and heterogeneous random forest, which basically meets the needs of the financial industry in many application scenarios such as risk assessment, anti-money laundering, investment advisory, investment research, credit, insurance and compliance; and in the top presentation layer, the platform provides a series of user interaction and task management capabilities in order to facilitate users to coordinate AI training tasks more effectively.

In addition to the capabilities provided by the traditional federated learning platform, the innovative features introduced by the Hive Federated Learning Platform include a federated learning-based medical imaging data platform, a user profiling and personalized recommender system, and a dynamic auto insurance pricing model system. These features, which are deeply integrated with specific applications in the financial industry, can help users achieve a lower learning curve and faster business integration.

Application Scenarios of the Platform

Currently, the Hive Federated Learning Platform is widely used in a range of business scenarios, all of which have achieved good results. Taking the platform's application in the insurance industry as an example; traditionally, when a customer buys insurance, only basic information such as the customer's age and gender are used to determine the premium amount. In today's digital era, the size and feature dimension of user data have increased tremendously. In the case of health insurance, for example, business systems can improve the accuracy of the policyholder's health assessment if they can use massive amounts of medical records, family history data, and more to make AI predictions and obtain more detailed health assessment.

However, there is no doubt that medical records and medical history are absolutely private data that should never be disclosed and need to be strictly protected in each healthcare organization. Now, with the introduction of the Hive Federated Learning Platform combined with Intel® SGX, companies can carry out AI training on insurance pricing model without having access to user data. And feedback from the front line deployment has shown a significant improvement in the effectiveness of the personalized insurance pricing.

In addition to its application in the personalized insurance pricing system, the Hive Federated Learning Platform has also achieved good results in the following scenarios:

- **Intelligent Internet of Vehicles Big Data Analysis**

The Hive Federated Learning Platform can aggregate customer profiles from different industries (such as insurance companies, operating fleets and used car evaluation platforms) while complying with data privacy laws and regulations, and provide automakers with personalized big data analytics services (for example analysis results/reports), thereby developing a deep insight into their own data. The actual data can also be used to validate, evaluate, and predict the business situation of automakers, so as to create more value for them in a more comprehensive way. Specifically, the Hive Federated Learning Platform can be used to analysis the data in terms of economy, environmental suitability, reliability, and safety to help the R&D and quality assurance departments of automakers make informed decision.

- **Accurate and Complete Social Credit Rating System**

It is an irreversible trend that the offline and online individual credit rating systems will be connected and further improved. By aggregating the accumulated data from daily life, finance, social network, e-commerce, transportation, and authority, the Hive Federated Learning Platform is able to expand the data dimensionality of private credit rating agencies, and profile a more comprehensive picture of each person's credit, thus building an accurate social credit rating system.

- **Clinical Decision Support System**

The federated image analysis and recognition capabilities provided by the Hive Federated Learning Platform allows the primary medical institutions to build an integrated medical service platform, where the medical image data (X-ray, CT, MRI) can be recognized, or medical literature data can be exploited to build a medical expert database, providing intelligent solutions for medical personnel, and improving their productivity and quality of treatment.

- **Intelligent Voice Management System**

The Hive Federated Learning Platform can support the call recording application in the financial industry by providing all relevant stakeholders with voice recognition, call transcription, and intelligent voice data analysis capability. The optimized voice recognition system can identify, count, and analyze a large number of call records, which allows the small and micro-sized financial institutions to understand the composition of different business calls in the shortest possible time, carry out more efficient and accurate intelligent quality assurance and valuable information extraction, locate the causes of customer complaints, customer attritions, and abnormal calls predict business trends, and identify potential customers.

Conclusion

With the help of the advanced federated learning method as well as the advanced hardware and software products such as Intel® SGX and Intel processors, financial enterprises are now able to effectively address the challenge of secure collaboration among multiple data sources in their advanced AI applications. In this regard, to meet the needs of financial enterprises in application scenarios such as risk assessment, anti-money laundering, investment advisory, investment research, credit, insurance and compliance, Intel and many financial partners, by using the federated learning-based AI applications, build more effective risk control, marketing and management models in order to effectively identify credit card fraud, overdue loans, financial fraud and other potential financial risks, thereby greatly reducing the business risks of financial enterprises and exemplifying the useful implementation of the federated learning in the financial industry.

In the future, Intel also plans to carry out in-depth technical cooperation with more financial enterprises by utilizing its advanced technologies to drive the secure operation and efficient transformation of data resources in federated learning, eliminate data silos on the premise of ensuring data security, and promote the rapid development and application of federated learning.

Utilize Advanced Memory Products and Innovative Algorithm Models to Facilitate the Implementation of High-availability Low-TCO AI-based Financial Solutions

AI Implementation Solution based on Financial Data Characteristics

Financial data with high-dimensional features requires better memory solution

Financial data with high-dimensional features

As one of the three drivers of AI, data plays a critical role in AI applications and solutions. In the process of implementing AI applications and solutions, the ability to represent data features in different industries will place different demands on the algorithmic computing power. In the case of social network scenario, for example, interaction data and user identity data have strong data relevance and single data dimension; in the e-commerce scenario, data mainly includes sales records, product data and user purchasing behavior, characterized by less data dimensions and relatively complex data features; in the industrial manufacturing field, data mainly includes process data such as logs, with strong time series nature and relatively few features.

In the financial industry, the data used and processed by banks, insurance companies and other financial institutions also has its own characteristics. Taking a typical prediction system as an example, as shown in Figure 2-6-1, a commercial bank needs to predict the credit risk by predicting the house price trend of City A in a future period of time. To do this, the basic process is to collect a sample dataset, use the machine learning/deep learning method to train such dataset and obtain a model, use the model and a prediction dataset to infer the result, compare the prediction result with the actual result, and optimize the model by means of model self-learning.

In the above process, the ability to represent features of the dataset (both the sample dataset and the prediction dataset) is critical to the performance of the machine learning/deep learning method. In the financial industry, the data features can usually be represented from both the macro and micro perspectives. In the case of a personalized recommender system, for example, the attributes of items and systems, such as price and time period, are common to all users (that are, macro descriptions), while users' own attributes, such as age and gender, are specific features (that are, micro characteristics). Again, for anti-fraud services, the attributes of the banking platform are macro-level common features, while users' own transaction and account information are micro-level specific features. When all of the above user data features are acquired and derived, we will obtain an extremely high number of data dimensions.

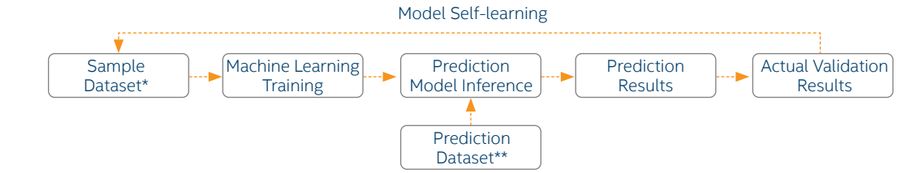
Therefore, when building an AI learning method for the financial data with high-dimensional features, the training efficiency is usually not as good as desired. The reason is that, although the large number of user samples in the dataset provides a large-scale specific features at the micro level, which can represent individual behaviors more accurately and better, the huge amount of data samples requires higher computing and storage performance when building machine learning/deep learning models. As a result, when dealing with data with high-dimensional features, the financial AI applications need to not only select innovative and unique algorithms, but also adopt a more advanced hardware infrastructure platform to provide powerful performance.

■ Financial high-dimensional data model demands for high performance memory solution

It is well known that the larger the training sample size and the higher the data dimension of a machine learning/deep learning model, the more information it contains, but the computing and processing will become more complex accordingly, which also raises the bar even higher on the performance of the underlying hardware infrastructure, especially memory. For high-dimensional models, they will build a giant pyramidal data matrix with underlying data dimensions that can be in the hundreds of millions or even billions. Therefore, on the one hand, they have a higher demand for parallel, real-time computing capability of the processor platform, which requires processors with higher clock frequency, more cores/threads and optimized micro-architecture; on the other hand, these high-dimensional data models also have an urgent need for high-capacity, high-performance memory.

Data modeling analysis shows that, when the dimensionality of a data model is in the millions, the model file size is usually in the GB level, and when it is in the billions, the file size can be as large as TB level. And when the AI system performs model computing and updates, it will generate a large amount of intermediate results for the iterative process.

In traditional system design, these iterative tasks that require low latency and high throughput are carried out by the high-performance Dynamic Random Access Memory (DRAM). But as the dimensionality of financial data continue to grow, the use of DRAM becomes less affordable.



*Sample Dataset

Features	City	Location	Area	# of Stories	...	Price
	City A	2nd Ring	120	12	...	9,000,000
Samples for Learning	City A	3rd Ring	95	4	...	7,200,000
	...	...	...	...	...	...

** Prediction Dataset

Features	City	Location	Area	# of Stories	...	Price
	City A	2nd Ring	100	8	...	?
Prediction Object	City A	2nd Ring	90	6	...	?
	...	...	...	...	...	?

Figure 2-6-1 AI-based Financial Prediction System

The above recommender system, for example, has hundreds of original features, and after further derivation, the number of features may reach to nearly 100 million. As a result, when using a single recommender system, its in-memory database will require hundreds of GB of memory, and when an enterprise uses more than ten recommender systems at the same time, the memory capacity required by the in-memory database will be as large as several TBs. Such huge amount of memory is not only a heavy cost burden on financial enterprises, but also requires them to deploy a larger cluster to accommodate the large-capacity DRAM, which puts tremendous pressure on their IT operation and management, and challenge the implementation of AI solutions.

To address this challenge, Intel is working with partners such as 4Paradigm to build several exemplary AI-based financial applications for high-dimensional data by combining a range of innovative high-dimensional feature algorithms with technologies such as RTiDB real-time feature database, based on Intel® Optane™ Persistent Memory and other advanced Intel hardware, software and technologies.

Advanced Memory Products Combined with Innovative Algorithms to Reduce Costs and Improve Efficiency for AI-based Financial Applications

■ Innovative algorithms designed for data with high-dimensional features

When dealing with high-dimensional feature datasets, traditional machine learning or deep learning algorithms commonly used in the financial industry, such as Logistics Regression (LR), Gradient Boosting Machine (GBM) and Deep Neural Network (DNN), often have limitations such as large memory usage, high computational complexity and low training efficiency. In order to enable Intel® Optane™ Persistent Memory to exert greater performance in such scenarios, Intel, in conjunction with 4Paradigm, has optimized a number of model algorithms for data with high-dimensional features, including:

- **HD-LR Linear Model:** It uses the FTRL (Follow The Regularized Leader) algorithm to iteratively solve the model, which allows for faster solution speed and supports for regularization operator, making it ideal for very large datasets with large numbers of sparse features;
- **HD-GBM (Gradient Boosting Machine):** An HD-Linear model is added. When dealing with high-dimensional sparse feature data, the decision tree sub-model and the HD-Linear sub-model can be boosted iteratively and alternately, so that a better training performance can be obtained by improving the interaction between these two sub-models.
- **HD-DSN (Deep Sparse Network) Algorithm**: Compared to the traditional DNN algorithm, the HD-DSN algorithm not only carries out high-level feature abstraction on a variety of ultra-high dimensional sparse data, but also automatically learns the embedded expressions. The number of data dimensions supported by the HD-DSN algorithm now reaches the trillion level.
- **HD-He-Treenet (Hyper-dimension Ensemble Tree Net) Model:** When dealing with datasets with strong time series nature, the HD-He-Treenet model can embed the high-dimensional, sparse discrete features by using a series of statistical-based algorithms, thereby significantly reducing the feature dimensionality and transforming the sparse features into dense ones without losing the information contained in the original features. Meanwhile, after embedding into the time series, the transformed features will be further enriched, which further improves the training performance.

■ Intel® Optane™ Persistent Memory for High-Dimensional Features

In traditional IT systems of financial enterprise, the Hard Disk Drives (HDDs) or Solid State Drives (SSDs) are usually used to store the large-scale intermediate business data. However, as the data features to be processed by the AI-based financial applications such as the recommender system are growing exponentially, and the data access speed needs to be improved dramatically to improve the real-time processing performance, one of the most effective ways to address these challenges is to store more data closer to the processor (as shown in Figure 2-6-2, the higher levels are closer to the processor).

In recent years, more and more data is stored into the DRAM memory, and various in-memory databases have become an indispensable part for processing "hot data" in AI applications. But the use of large-scale DRAM memory will increase the procurement and maintenance costs, and due to the volatile nature of DRAM memory, the working process will take more time to reload data when the system experiences downtime and other problems.

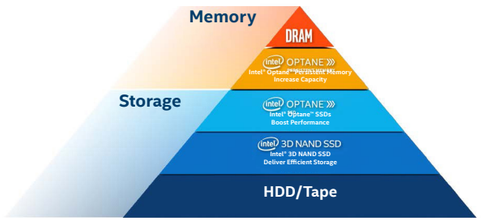


Figure 2-6-2 Memory-level Performance and High-capacity Persistent Storage Capability from Intel® Optane™ Persistent Memory[22]

To address the high-capacity, low-latency and high-performance data load/read/write required by the high-dimensional feature model processing, Intel provides a highly available, low-TCO Persistent Memory Module (PMM) solution by utilizing the Intel® Optane™ technology with the innovative algorithms described in the previous section to meet the high performance and high-capacity requirements of AI applications in the financial industry.

Built on a unique 3D XPoint™ storage medium, Intel® Optane™ Persistent Memory combines memory-class performance with high-capacity persistent storage capabilities in a dense, transistor-less and stackable design that allows memory cells to be individually addressed. When using with other Intel advanced system memory and storage controllers, interface hardware, and software enhancements, its read and write performance and access latency are comparable to those of DRAM memory.

[22] Image cited from the Intel website: https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology/reimagine-memory-storage-in-the-data-center.html

When operating in Memory Mode, as shown in Figure 2-6-3, Intel® Optane™ Persistent Memory is compatible with DRAM memory after plugging into the standard Dual-Inline Memory-Modules (DIMM) memory slots. However, unlike DRAM memory, Intel® Optane™ Persistent Memory also offers two important features that are game changer for memory and storage: persistence and high density. The persistence means that data will be preserved even in case of a power failure or reboot, while the high density refers to a capacity of up to 512GB per DIMM slot, which is several times the maximum density of current DRAM memory.

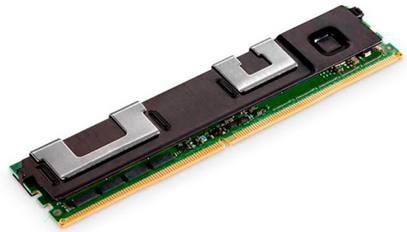


Figure 2-6-3 Intel® Optane™ Persistent Memory[23]

By virtue of these two features, on the one hand, the in-memory database deployed by financial enterprises can be preserved without reloading in case of a system maintenance or unexpected downtime, which not only avoids a series of problems caused by data loss, but also makes sure that the training and inference of the AI system will no longer be affected by accidents and maintain strong consistency, while the system recovery time can be reduced to a few seconds; on the other hand, the lower cost per GB of memory gives financial enterprises the flexibility to deploy more large-capacity in-memory databases in their data center, equip virtual machines with more memory, or further increase the density of virtual machines, thereby comprehensively improving the productivity of AI applications for financial data and enabling them to more easily process the high-dimensional data features.

■ Intel® PMDK for Financial Data

In addition to advanced hardware products, Intel also provides a programming model and environment for Intel® Optane™ Persistent Memory through the Intel® Persistent Memory Development Kit (PMDK), which works with the high-performance in-memory database to build a persistent memory solution for financial data that allows financial enterprises to easily transform high-dimensional data into persistent in-memory data structures for subsequent model training and inference.

In this solution, since the data lifecycle of persistent memory is often longer than the lifecycle of a program process, traditional programming models including memory allocation and management and virtual address-based pointers, built on volatile memory (e.g., DRAM memory), will no longer be applicable. The Intel® PMDK introduces a new persistent memory programming model, which uses the latest libpmemobj-cpp to create the persistent data structure and builds the KV (key-value) storage engine, thus facilitating users to smoothly migrate existing programs built on the traditional memory model into the persistent memory with minimal code changes, and ensuring that all data is kept consistent under any circumstances (such as reboots and system crashes due to hardware and software errors). In the following section, we will briefly describe the design and implementation of the PmemStore-KV storage engine and its integration with the high-performance RTiDB real-time feature database.

Design and implementation of the underlying data structure
The underlying data structure determines how the key is organized, or how the value is indexed (**index**), which will directly affect the performance of the KV engine in different workloads. For example, when there are more point queries, it is more appropriate to use **hash-based index**, and for range search, it is more suitable to use **b+tree** or **skiplist**.

The debug and data consistency test on the persistent data structures is one of the most important challenges that affect the wide-spread application of the persistent memory programming, and the industry is still exploring the right answer to this question. Now, several widely used tools, such as **pmemcheck** (in **valgrind**) and pmreorder, may be helpful in checking the errors of Intel® PMDK programs.

KV engine architecture design and implementation
Based on the **key → pmem_pointer** and **pmem_pointer → value** architecture, Intel® PMDK can enable users to implement a basic local KV engine, where managing value will be critical to the performance. Therefore, the solution adopts the following design:

- Persistent memory management based on Intel® PMDK, which can effectively manage the allocation and recycling of value space.
- Several data consistency policies with different performance available for use, allowing users to make trade-offs based on their performance requirements.

High performance RTiDB real-time feature database
The traditional relational databases or in-memory databases, such as MySQL and Redis, are not designed for extracting the time series features, which making them unable to extract high-dimensional time series feature at a speed of milliseconds and handle the I/O spikes resulting from the high-dimensional feature extraction. And even some specific time series databases are not able to meet the performance requirement of a financial hard real-time scenario with TP99 (i.e., able to meet the minimum latency required by 99% of requests) of 5 milliseconds.

RTiDB, introduced by 4Paradigm, is designed by using the sorted core data structure, non-blocking execution between read and write, and less transactions, and has a built-in high-dimensional feature extraction engine Feature Extractor, offering tremendous advantages in the processing of high-dimensional feature data. Compared to typical in-memory databases, it achieved 5-10x improvement in time series extraction performance, and 2x improvement in PUT performance[24]. Therefore, it is very suitable for anti-fraud, anti-money laundering, marketing, recommendation and other scenarios in the financial industry.

[23] Image cited from the Intel website: https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology/reimagine-memory-storage-in-the-data-center.html

[24] Performance data cited from 4Paradigm AI Data Platform Architecture

As can be seen from the system design of RTiDB, its core Tablet Server module consists of a memory-based storage engine and a unique memory recovery model. The former requires high-capacity and high-performance memory, so the relatively high price and the relatively limited memory density per slot of traditional DRAM memory is undoubtedly the bottleneck to its performance. And the latter relies on the persistent in-memory data structures. Therefore, the key to maximizing the performance of RTiDB is to find a memory product that offers the right combination of high capacity and persistence, and Intel® Optane™ Persistent Memory is certainly an advanced memory product that meets this need.

The RTiDB real-time feature database is now well integrated with Intel® Optane™ Persistent Memory with the help of the KV engine architecture provided by Intel® PMDK. Intel is currently working with 4Paradigm to explore more optimized high-performance solutions. Although the DRAM memory-based RTiDB still has a performance advantage over the Intel® Optane™ Persistent Memory-based RTiDB, by disabling synchronization of the snapshot/binlog to disk, Intel® Optane™ Persistent Memory-based RTIDB has optimized the latency of TP9999/TP99999 (i.e., able to meet the minimum latency required by 99.99%/99.999% of requests) write operations.

For more information on Intel® PMDK, please visit https://pmem.io/pmdk/

■ Example of Rapid Database Recovery using PMDK

First, the system administrators and application developers can obtain the PMDK toolkit by visiting **https://pmem.io/**, and then install the toolkit as follows:

```
# install pmdk

sudo yum install ndctl-devel daxctl-devel
git clone https://github.com/pmem/pmdk
cd pmdk
git checkout tags/1.8
make -j4
sudo make install prefix=/usr/local
```

The **libpmemobj-cpp** library will also need to be installed:

```
# install libpmemobj-cpp

git clone https://github.com/pmem/libpmemobj-cpp
cd libpmemobj-cpp
git checkout tags/1.9
mkdir build && cd build
cmake .. -DTBB_DIR=$TBBROOT/cmake -DCMAKE_INSTALL_PREFIX=/usr/local -
DUSE_TBB=1
make -j4
sudo make install
```

Include the necessary header files

```
#include "libpmemobj++/container/concurrent_hash_map.hpp"
#include "libpmemobj++/p.hpp"
#include "libpmemobj++/persistent_ptr.hpp"
#include "libpmemobj++/pool.hpp"
#include "libpmemobj++/utils.hpp"
#include "libpmemobj++/make_persistent.hpp"
#include "libpmemobj++/make_persistent_array.hpp"
#include "libpmemobj++/transaction.hpp"
```

To perform initialization, you need to create or open a pmempool. The sample codes are as follows:

```
try {
    if (!rtidb::base::IsExists(FLAGS_pmem_pool_file)) {
        LOG(INFO, "pmem pool file does not exist, start to create a new one");
        pop_ = pool<::rtidb::storage::PmemRoot>::create(FLAGS_pmem_pool_file,
                PMEM_LAYOUT, FLAGS_pmem_pool_size,S_IWUSR | S_IRUSR);
        LOG(INFO, "new pmem pool file created, size = %llu", FLAGS_pmem_pool_size);
    } else {
        LOG(INFO, "pmem pool file found, use the existing one");
        pop_ = pool<::rtidb::storage::PmemRoot>::open(FLAGS_pmem_pool_file, PMEM_LAYOUT);
    }
} catch (pmem::pool_error &e) {
    LOG(ERROR, "failed to open/create pmem pool file");
    return false;
}
```

PMemRoot is saved in the root location of pmempool and is the entry point for finding all RTiDB table data. You can use the persistent concurrent hash map provided by libpmemobj-cpp to create a mapping table from table id to tabledata. The sample codes are as follows:

```
tabledata 的 mapping table
class PmemRoot {
public:
    using hashmap_type = concurrent_hash_map<p<uint64_t>, persistent_ptr<PmemTableData>>;
    persistent_ptr<hashmap_type> pmem_tables;
};
```

When creating a new RTiDB table, you need to check if the mapping table in root exists first, and if it doesn't, you need to create it beforehand. Then, initialize the RTiDB table with the necessary data (PmemTableData) and insert it into the mapping table. The sample codes are as follows:

```
if (pop_.root()->pmem_tables == nullptr) {
    LOG(INFO, "pop_.root()->pmem_tables is null, start to create one");
    try {
        pmem::obj::transaction::run(pop_, [&] {
            pop_.root()->pmem_tables = make_persistent<PmemRoot::hashmap_type>();
        });
    } catch (const pmem::transaction_error &e) {
        LOG(ERROR, "failed to allocation pmem for root->pmem_tables, %s", e.what());
        return false;
    }
} else {
    LOG(INFO, "pop_.root()->pmem_tables not null, start to initialize");
    pop_.root()->pmem_tables->runtime_initialize();
}
PmemRoot::hashmap_type::accessor acc;
bool res = pop_.root()->pmem_tables->find(acc, tid_pid);
if (res) {
    LOG(WARNING, "pmem tid %u pid %u exist", id_, pid_);
    return false;
} else
{
    LOG(INFO, "start to create pmem_table_data");
    try {
        pmem::obj::transaction::run(pop_, [&] {
            meta_ = make_persistent<PmemTableMeta>();
            persistent_ptr<persistent_ptr<PmemSegmentData>[]> segs_data
                = make_persistent<persistent_ptr<PmemSegmentData>[]>(idx_cnt_*seg_cnt_);
            pmem_table_data = make_persistent<PmemTableData>(meta_, segs_data);
            for (uint32_t i = 0; i < idx_cnt_; i++) {
                for (uint32_t j = 0; j < seg_cnt_; j++) {
                    segs_data[i * seg_cnt_ + j] = segments_[i][j]->Init();
                }
            }
        });
    } catch (const pmem::transaction_error &e) {
        LOG(ERROR, "failed to allocation pmem for root->pmem_tables, %s", e.what());
        return false;
    }
    LOG(INFO, "create pmem_table_data success, start to insert to pmem_tables");
    pop_.root()->pmem_tables->insert(PmemRoot::hashmap_type::value_type(tid_pid, pmem_tal
}
```

When opening an existing RTiDB table, you also need to check if the mapping table in root exists first, and if it doesn't, you need to create it beforehand. Then, in the mapping table, find the necessary data (PmemTableData) for the RTiDB table, which will be used for instant recovery of the table. The sample codes are as follows:

```
if (pop_.root()->pmem_tables == nullptr) {
    try {
        pmem::obj::transaction::run(pop_, [&] {
            pop_.root()->pmem_tables = make_persistent<PmemRoot::hashmap_type>();
        });
    } catch (const pmem::transaction_error &e) {
        LOG(ERROR, "failed to allocation pmem for root->pmem_tables, %s", e.what());
        return false;
    }
} else {
    pop_.root()->pmem_tables->runtime_initialize();
}
PmemRoot::hashmap_type::accessor acc;
bool res = pop_.root()->pmem_tables->find(acc, tid_pid);
if (res) {
    pmem_table_data = acc->second;
    meta_ = pmem_table_data->meta_;
    for (uint32_t i = 0; i < idx_cnt_; i++) {
        for (uint32_t j = 0; j < seg_cnt_; j++) {
            segments_[i][j]->Load(pmem_table_data->segs_data_[i * seg_cnt_ + j]);
        }
    }
} else
{
    PDLOG(WARNING, "pmem table tid %u pid %u does not exist", id_, pid_);
    return false;
}
```

■ Intel® Optane™ Persistent Memory Performance and TCO Benchmarking

To validate the performance and TCO improvements that Intel® Optane™ Persistent Memory can bring to AI scenarios in the financial industry after it is integrated with RTiDB and other products and technologies, Intel and 4Paradigm conducted a benchmarking between DRAM memory and Intel® Optane™ Persistent Memory on performance and TCO. The configuration of the benchmarking is as follows:

System Configuration		
Processor	2S Intel® Xeon® Platinum 8280L Processor	
Processor Frequency	2.70GHz (Turbo Boost @ 4.0GHz)	
L1/L2/L3 Caches	1.75MB/28MB/38.5MB	
DRAM Memory	384GB (12*32GB DDR4 2666MHz)	
Intel® Optane™ Persistent Memory	2TB (8*256GB 2666MHz)	
Storage	750GB Intel® Optane™ SSD DC P4800X	
Operating System	CentOS-7.6 (Kernel 5.1.9-1.el7)	
Intel® Optane™ Persistent Memory Firmware	01.02.00.5375	
RTiDB Configuration		
DRAM Memory	Volatile Skiplist@DRAM Memory	Logs stored in Intel® Optane™ SSDs
Intel® Optane™ Persistent Memory	Persistent Skiplist@ Intel® Optane™ Persistent Memory (AD Mode)	No additional storage for logs

Configuration of Intel® Optane™ Persistent Memory Benchmarking

Benchmarking Procedure:

1. Data preparation

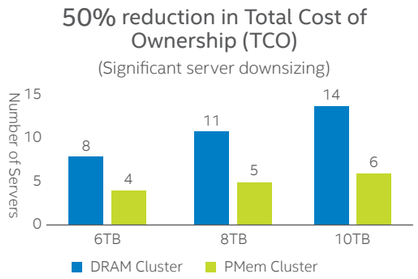
```
##Schema
ttl_type: kAbsoluteTime
ttl: 0
partition_num: 8
replica_num: 3
columns:
-
name: "card"
type: "string"
add_ts_idx: true
-
name: "mcc"
type: "string"
-
name: "amt"
type: "string"
-
name: "num"
type: "int64"
-
name: "coll"
type: "double"
-
name: "ts1"
type: "int64"
is_ts_col: true
-
name: "ts2"
type: "int64"
is_ts_col: true
column_keys:
-
index_name: "combined_key1"
col_name: ["card"]
ts_name: ["ts1","ts2"]
-
index_name: "combined_key2"
col_name: ["mcc","amt"]
ts_name: ["ts1"]
-
index_name: "combined_key3"
col_name: ["num"]
ts_name: ["ts1"]
```

2. **get & scan** operations

- **get** and **scan** merge operations;
- Operate ***key*** on 50 entries;
- Stress test: 1-128 threads in an hour.

3. **put** operations

- In case that the original table exists, create a new table with a random number of key-value pairs.
- Stress test: 128 threads in 10 minutes.



Data Size	Number of DRAM Servers	Number of PMem Servers
6TB	8	4
8TB	11	5
10TB	14	6

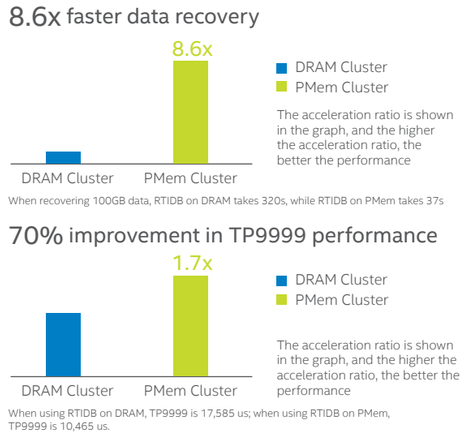


Figure 2-6-4 Benchmarking between DRAM Memory and Intel® Optane™ Persistent Memory on Performance and TCO

As shown in Figure 2-6-4, the benchmarking results show that, when using different data sizes, compared to DRAM memory, Intel® Optane™ Persistent Memory can reduce the overall TCO by 50%, improve the data recovery speed by 8.6 times, and increase the TP9999 performance (i.e., able to meet the minimum latency required by 99.99% of requests) by 70%.

For more details on the benchmarking, see 4Paradigm's internal report Sigmod2020 Paper Experimental Results: https://wiki.4paradigm.com/display/PlatformRD/Sigmod2020+Paper+Experimental+Results

Application Scenarios

By combining innovative algorithms, RTiDB and Intel® Optane™ Persistent Memory, the AI solution for high-dimensional feature data has been implemented in a variety of financial scenarios. These include:

Transaction fraud detection: On the one hand, by integrating the anti-fraud scenarios in banks and payment institutions, the new solution can build ultra-high-dimensional machine learning models to implement the fraud prevention and control. On the other hand, the solution can also be used with the real-time information processing and online services to provide real-time transaction anti-fraud capabilities for banks, e-commerce and payment institutions. Compared with the traditional expert system, the detection accuracy rate is several times higher than the original rules, and the fraud transaction can be blocked in real time in millisecond.

Credit risk control: The solution can be used with knowledge graph, Natural Language Processing (NLP) and other technologies to provide risk control at each key step in the whole credit life cycle. It covers multiple business scenarios such as risk alert, information verification, human-machine gaming, fraud detection, post-loan management, loan collection & alert and connection reestablishment.

Supply chain finance: Through in-depth analysis and real-time computing and monitoring of supply chain data, enterprise data, market data and financial institutions' data, the solution enables financial institutions to timely know enterprises' capital needs, and accurately understand enterprises' operating and risk conditions, so as to provide them with better services; on the other hand, the solution can also integrate financial services into the supply chain transaction management process, as well as the production and operation of enterprises, thereby making it simple and easy for small and medium-sized enterprises to access financial services.

Intelligent customer service: the solution can use massive data to build a dialogue model, which can be used to enable self-learning by combining multiple rounds of dialogue with real-time feedback, accurately identify user intent, support text, voice and picture interaction, and perform multi-domain semantic analysis and multiple forms of Q&A, thereby transforming the Passive Q&A to the Active Q&A to improve user experience.

Recommended Hardware and Software Configuration

The above Intel® Optane™ Persistent Memory-based machine learning solution can be built on the following Intel® architecture platform, with the following environment configurations:

Hardware Configuration

Name	Specification
Processor	2S Intel® Xeon® Platinum 8280L Processor or higher
Base Frequency	2.70GHz (Turbo Boost @ 4.0GHz)
Cores/Threads	28/56
HT	On
Turbo	On
Memory	2TB (8*256GB 2666MHz) Intel® Optane™ Persistent Memory or higher
Storage	750GB Intel® Optane™ SSD DC P4800X or higher

Software Configuration

Name	Specification
Operating System	CentOS-7 or other Linux systems
Linux Kernel	5.1.9-1.el7
Workloads	4Paradigm Sage with built-in feature database RTiDB
Compiler	GCC 5.4
Library	Jdk (jdk-8u121-linux-x64.tar.gz) Zookeeper3.4

Technology Outlook

The in-memory databases are heading toward wider deployment in financial enterprises as financial services demand better real-time, interactive, and AI application capabilities. As an innovative memory technology, Intel® Optane™ Persistent Memory products, by virtue of their more cost-effective capacity and good support for data persistence, are enabling financial enterprises to more effectively implement a range of machine learning/deep learning methods around in-memory databases for high-dimensional time series data, providing AI capabilities to effectively support financial services.

In the future, more users in the financial industry are planning to integrate Intel® Optane™ Persistent Memory with more technologies and business scenarios to create more effective advanced data analysis and AI applications. For example, with the advent of the 5G era, ultra-low latency, ultra-bandwidth network experience also enables financial services to provide a comparable fast response for data analysis and AI applications. But the responsiveness of data processing capabilities traditionally built in data centers, or in the cloud, will undoubtedly be affected by network quality, bandwidth, and other factors.

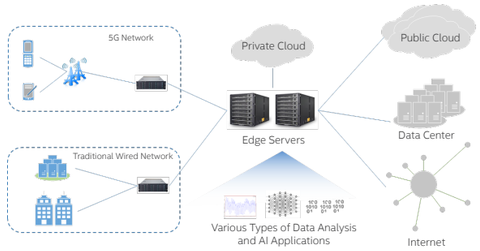


Figure 2-6-5 Architecture of Typical Edge Computing Node Deployment

To this end, an increasing number of financial enterprises are deploying the Mobile Edge Computing (MEC) nodes at the network edge to address these challenges. A typical MEC node deployment architecture is shown in Figure 2-6-5, where the edge servers enable enterprises to deploy the low-latency, high-bandwidth, and localized services, especially applications that require large amounts of data processing, storage, and transport, to the edge of the network, making these services more user-aware and user-experienced.

Meanwhile, the edge servers are also subject to greater performance, capacity, and availability challenges, and under this circumstance, Intel® Optane™ Persistent Memory is definitely superior to traditional edge servers equipped with DRAM memory. On the one hand, the largest capacity of a single Intel® Optane™ Persistent Memory module now reaches 512GB, while that of the most common DRAM memory is only 32GB, which indicates a huge difference in memory density. Larger memory means that financial enterprises can deploy more virtual machines or service processes in edge servers and reduce the number of servers, lowering TCO. On the other hand, Intel® Optane™ Persistent Memory with built-in AES 256 bit hardware encryption technology provides greater data security at the edge for financial services.

To further help users improve their data processing capability, Intel has introduced the brand new 2nd Generation Intel® Optane™ Persistent Memory, as shown in Figure 2-6-6, which offers an average increase in memory bandwidth of 25%[25] over the previous generation, with up to 4.5TB of memory capacity per socket[26], and in some scenarios, allows users to reduce database boot time by up to 12.5 times[27]. As a result, it delivers users with a more powerful processing performance and a larger data processing space, providing a better data processing platform for the financial industry.

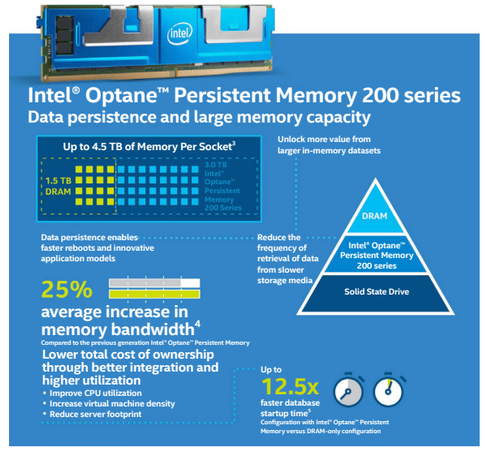


Figure 2-6-6 Brand New 2nd Generation Intel® Optane™ Persistent Memory[28]

[25] Data test configuration: baseline configuration: single node with one Intel® Xeon® 8280L 28C@2.7GHz processor on Neon City, single persistent memory module configuration (6 x 32GB DDR4 DRAM; 1 x {128GB, 256GB, 512GB} Intel® Optane™ Persistent Memory 100 series module, 15W), ucode 0x04002f00, for running Fedora29 kernel 5.1.18-200.fc29.x86_64 and MLC version 3.8 in App-Direct mode. Data source: 2020ww18_CPX_BPS_DI. Tested by Intel, on 27 April 2020. New configuration: single node with one Intel® Xeon® Pre-Production CPX6 28C@2.9GHz processor on Cooper City, single persistent memory module configuration (6 x 32GB DDR4 DRAM; 1 x {128GB, 256GB, 512GB} Intel® Optane™ Persistent Memory 200 Series module, 15W), ucode pre-production, for running Fedora29 kernel 5.1.18-200.fc29.x86_64 and MLC version 3.8 in App-Direct mode. Data source: 2020ww18_CPX__BPS_BG. Tested by Intel, on 31 March 2020.

[26] The configuration is 6 x 512 GB Intel® Optane™ Persistent Memory (3,072 GB) + 6 x 256 GB DDR4 DRAM (1,536 GB) = 4,608 GB total memory per socket.

[27] Data from test on 30 May 2018. SAP HANA simulated workload, SAP BW, edition for SAP HANA Standard Application Benchmark, Version 2 (30 May 2018). Baseline configuration for traditional DRAM: Lenovo ThinkSystem SR950 server with eight Intel® Xeon® Platinum 8176M processors (28 cores, 165 watts, 2.1 GHz). Total memory consists of 48 x 16 GB TruDDR4 2,666 MHz RDIMMs, and 5 x ThinkSystem 2.5" PM1633a 3.84TB capacity SAS 12GB hot swap SSDs for SAP HANA storage.The operating system is SUSE Linux Enterprise Server 12 SP3 and uses SAP HANA 2.0 SPS 03 with a 6 TB dataset. Average start time for all data finished after table preload for 10 iterations: 50 minutes.

[28] Figure cited from the Intel website: https://www.intel.cn/content/dam/www/public/cn/zh/images/20200715-turbocharge-your-ai-and-analytics-cn.jpg

Use Case: Application of 4Paradigm's Innovative Algorithm in a Commercial Bank

Background

The mobile banking APPs have become one of the most important online channels for commercial banks, so how to stand out from the crowd of APPs that provide basically the same functions? Providing personalized recommendations for different users in the pages of an APP according to the user profile is a great way to gain favor from customers.

A commercial bank, which has been working on the financial AI for many years, also wants to recommend suitable investment products to its customers at the featured areas of its mobile banking APP, thereby improving customer response rate and increasing revenue from investment product sales. Generally, financial enterprises will acquire customers' usage habits, usage tracks and behavioral heat maps by using event tracking in the APP in order to understand customers' preferences and concerns, and then choose appropriate featured areas to provide recommendations to customers.

Because the bank's mobile banking APP was developed a long time ago, there was no event tracking designed in the APP. In addition, the bank wants its recommender system to correlate with historical financial purchases from the bank's other channels to improve the recommendation success rate. Data analysis shows that, the huge amount of historical data accumulated by various channels has typical high-dimensional and sparse characteristics, which not only provides rich information, but also poses a challenge to the developing of the AI recommender solution.

By cooperating with 4Paradigm, Intel combined innovative algorithms, RTiDB and Intel® Optane™ Persistent Memory to train and infer on historical banking data with high-dimensional characteristics using machine learning methods, thereby accurately determining customers' investment needs and providing an appropriate recommender solution for the APP.

Solution Architecture and Deployment

The architecture of the new machine learning-based bank marketing recommender system is shown in Figure 2-6-7. In the solution, user data from the most recent year of the bank's integrated financial advisory system is selected as the sample set, which includes basic user information, investment product information, investment product purchase records, and user function information.

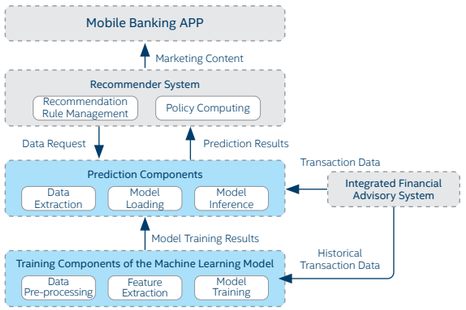


Figure 2-6-7 Marketing Recommender System Architecture of a Commercial Bank

First, the sample set undergoes data pre-processing, feature extraction, and model training in the training components of the machine learning model at the bottom of the architecture. The results obtained from model training are loaded into the prediction component to perform data extraction, model loading, and inference. The prediction results obtained by inference will go through the bank's recommender system and finally display on the mobile banking APP.

During this process, several steps, such as sample definition and sample construction, are key to the effectiveness of the solution. The sample definition is shown in Figure 2-6-8, in which, customer who purchased an investment product within 7 days of the start of the customer marketing date is identified as the positive sample customer. Otherwise, the customer is identified as the negative sample customer for that investment product.

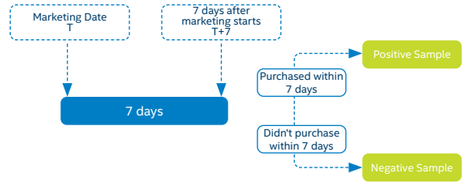


Figure 2-6-8 Sample Definition Process

In the sample construction process, as shown in Figure 2-6-9, the solution identifies the record of successful purchase of an investment product as a positive sample, and the marketing date is pushed back 7 days according to the time of purchase (to prevent pass-through, in conjunction with the recycling policy). So, a positive sample can be represented as: <cust_id,prd_id,mkt_time,1>.

The negative samples, on the other hand, use product as the dimension and are selected from customers who didn't purchase that product. In addition, the number of the negative samples is determined according to the positive to negative ratio of 1:50 (this ratio is determined according to the average response rate of investment product purchases, the cost of generating the sample, and the training run time), and the marketing time is randomly selected from the marketing time in the positive samples. So, a negative sample can be represented as: <cust_id,prd_id,mkt_time,0>

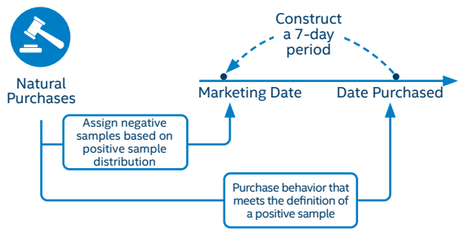


Figure 2-6-9 Sample Construction Process

The negative customer samples include two kinds of customer: customers who have never purchased an investment product, as well as customers who have never purchased Product A, but have purchased Product B. The former are used to determine whether customers will purchase an investment product or not, while the latter are identified as the negative customer samples of Product A, which is used to simulate customer's selectivity towards the product.

Next, the solution will perform feature extraction. Based on the analysis of the historical data in the integrated financial advisory system, the solution classifies the data into different feature categories such as user state, product attribute, user behavior and statistics, and under each category, dozens or hundreds of different feature sub-categories are also classified. This process will create nearly 100 million feature columns.

In the model training process, the solution filters **trans_time** and considers a 22-day data availability gap to obtain the positive sample list. At the same time, the solution also maintains the proportion of each investment product, while the negative customer samples are filtered from the customer base that has never purchased the product (i.e., the negative customer samples for Product A are filtered from the customer base that has never purchased Product A, and the negative customer samples for Product B are filtered from the customer base that has never purchased Product B). Then, the solution obtains a negative sample list in the recent two months, which is trained by the machine learning method to generate 17,021,921 samples. Finally, the solution selects a mass-market long-term product sold by the commercial bank to evaluate the prediction performance

Results

To verify whether the machine learning-based recommendation method is superior, a set of random recommendation methods (random group), a set of expert rule-based recommendation methods (expert rule group) and the machine learning method (machine learning group) are created to benchmark their performance. The design of the benchmarking between the three types of recommendation method is shown in Figure 2-6-10.

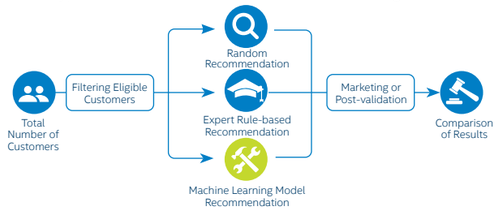


Figure 2-6-10 Design of Benchmarking between Three Types of Recommended Method

The benchmarking uses the A/B testing to compare the respective response rates of the expert rule group, random group and machine learning group. First, we filter out approximately 1.5 million private customers who meet the purchase criteria for a specific product. The customer list is then filtered by the expert rules group, the random group and the machine learning group respectively, and the SMS marketing is performed on eligible customers. Each of the three groups generates a list of 20,000 customers for real SMS marketing according to their own rules, and counts the results after the marketing is over.

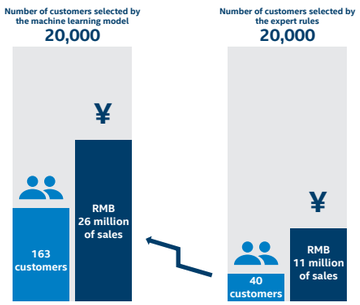


Figure 2-6-11 Comparison of Marketing Effectiveness of Different Recommendation Methods

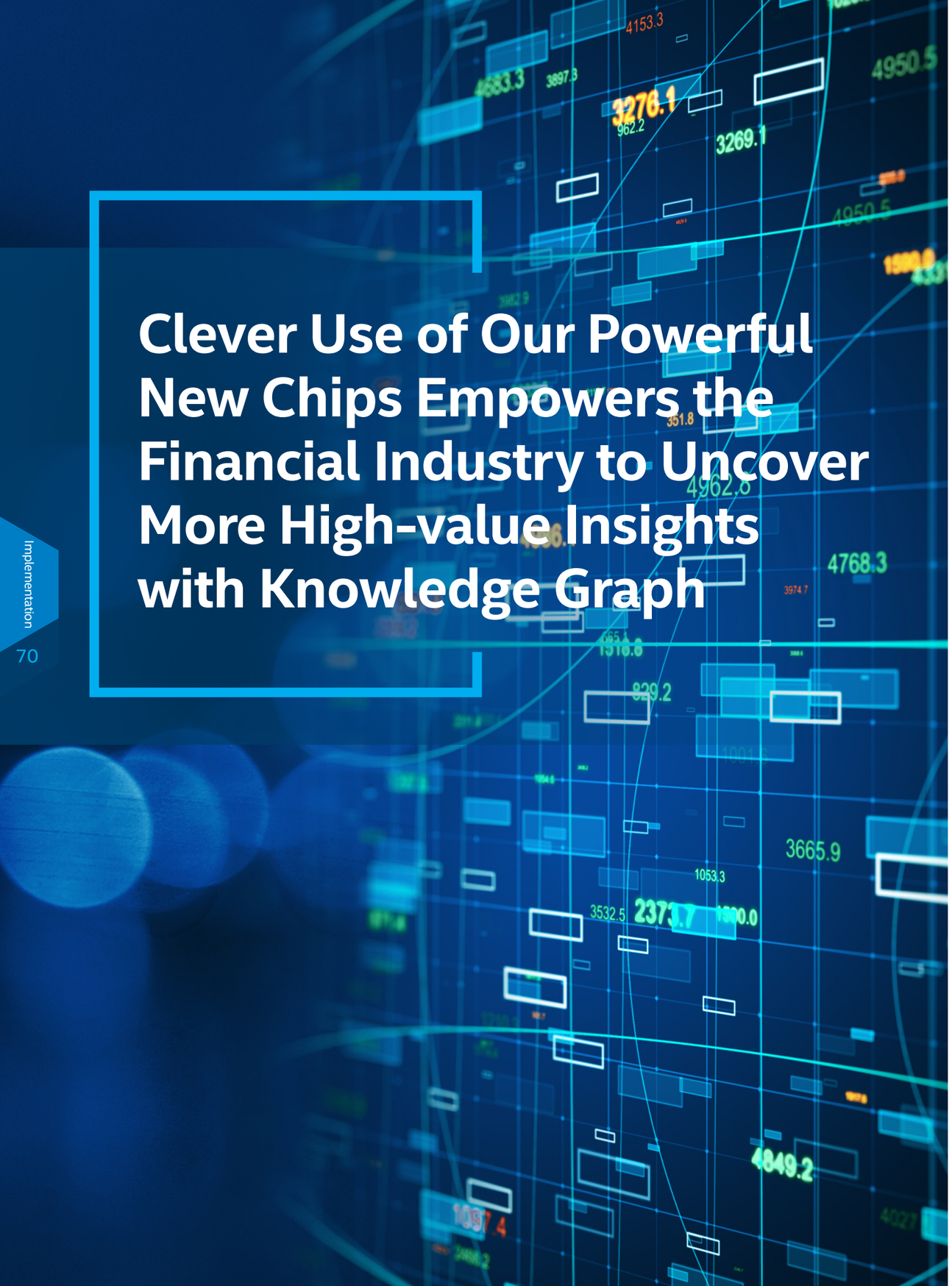
The test results are shown in Figure 2-6-11, where the machine learning group is far more effective in marketing than the expert rule and random groups. There were 163 customers who have made purchase in the machine learning group, or 29% of all purchasers, with sales totaling RMB 26.01 million, or 23% of all purchases; and 40 customers in the expert rule group made purchase, or 7% of all purchasers, with sales totaling RMB 11.07 million, or 10% of all purchases; and only 10 out of 20,000 customers in the random group made purchase, or 2% of all purchasers, with sales totaling RMB 870,000, or less than 1% of all purchases.

It is worth mentioning that the machine learning-based recommendation method can be trained, inferred and evaluated for effectiveness entirely online. Compared with the traditional expert rule-based recommendation method, the new solution allows users to understand the changes in the effectiveness of recommendation in real time, and make timely adjustments, which is undoubtedly a huge advantage for the financial industry.

Conclusion

Through a combination of innovative algorithms, RTiDB, and advanced Intel® Optane™ Persistent Memory products, Intel and its partners, including 4Paradigm, have built a real-time, online machine learning solution to address the characteristics of financial data, which enables financial enterprises to deploy AI applications to marketing recommendations, risk assessment, and other areas. In this solution, a series of innovative algorithms effectively reduce the computational complexity and time consumption caused by high-dimensional and sparse features, and improve the training performance, while the high-performance RTiDB real-time feature database with high-dimensional feature extraction engine provides powerful time series feature extraction performance. What's more, Intel® Optane™ Persistent Memory, by virtue of its large capacity, high performance and persistence, not only provides the larger memory required by the high-dimensional, sparse feature data processing, but also delivers fast data recovery speed that ensures the smooth online operation of the machine learning method.

In the future, Intel plans to work with more partners to conduct more technical discussions on the characteristics of financial data, and integrate our advanced hardware and software products with innovative algorithm models to facilitate the rapid implementation and application of high-availability, low-TCO AI-based financial solutions.



Clever Use of Our Powerful New Chips Empowers the Financial Industry to Uncover More High-value Insights with Knowledge Graph

Application of Knowledge Graph in Financial Industry

Knowledge Graph in Financial Industry

■ Introduction to the Knowledge Graph

As an important branch of AI, Knowledge Graph, with its accurate description of complex information relationships, is being increasingly deployed and applied in various industries. As shown in Figure 2-7-1, a knowledge graph is an AI system consisting of a set of entities, entity attributes, and the relationships between entities.

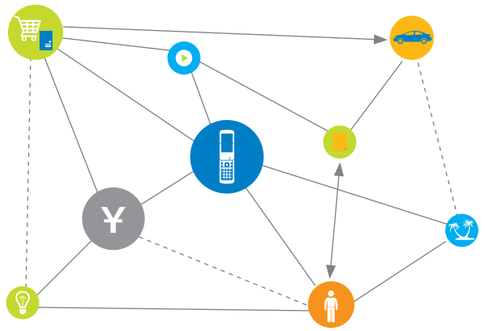


Figure 2-7-1 Knowledge Graph for Describing Complex Information Relationships

Through the knowledge graph, users can more accurately analyze the high-value information contained in massive data in the Internet era, as well as the interrelationships between the information, in order to obtain an in-depth and more effective solution. Nowadays, the knowledge graph systems have been used successfully in a large number of applications in the fields of information retrieval, recommender systems, anti-fraud, public opinion monitoring, market segmentation, and social chains.

For example, in the information retrieval field, search engines can use the knowledge graph to accurately aggregate and match the relevant information, and deepen the understanding of keywords and semantic analysis of search intent to improve search efficiency; in the recommender system, the knowledge graph can be used as an auxiliary information tool, integrated in e-commerce and other scenarios, to accurately match users' purchase intent and candidate products, providing users with more accurate recommendations; and in the social network, the knowledge graph provides a visual representation of relationships, optimized friend recommendation experience, and preference aggregation.

■ Building Knowledge Graph System for the Financial Industry

In the financial industry, the knowledge graph has also been widely used. For example, in the field of financial risk control, financial institutions can use the knowledge graph to analyze the relationship between entities, and then analyze the level of financial risk, so as to facilitate the formulation of countermeasures; in the field of financial marketing, financial institutions can use the knowledge graph to analyze and explore potential business opportunities and acquire more related high-quality customers, thereby enabling business personnel to access more customer acquisition models in addition to the traditional relationship marketing-based customer acquisition model, and expand the source of potential customers.

As shown in Figure 2-7-2, a knowledge graph system in the financial industry typically consists of data acquisition, data pre-processing, entity and entity relationship extraction, graph representation, data storage & management, task scheduling, and various internal and external service interfaces.

In the data acquisition step, on the one hand, the system collects public or internal data about companies and individuals from multiple internal and external data sources in a legally compliant manner. Public information includes company-related information and various public opinions and news obtained from the Internet, while internal information includes internal guarantee data and collateral data provided by various IT systems in financial institutions, which require efficient and sophisticated NLP technology for data acquisition. On the other hand, the system also needs to access a large amount of non-textual data, such as information existing in images and videos, which requires the system to have powerful Optical Character Recognition (OCR) capability.

In the subsequent data pre-processing module, the system first removes irregular, erroneous, and duplicate data in the data cleansing step, and then merges and aligns data from different sources, both internal and external, under the same entity through data fusion.

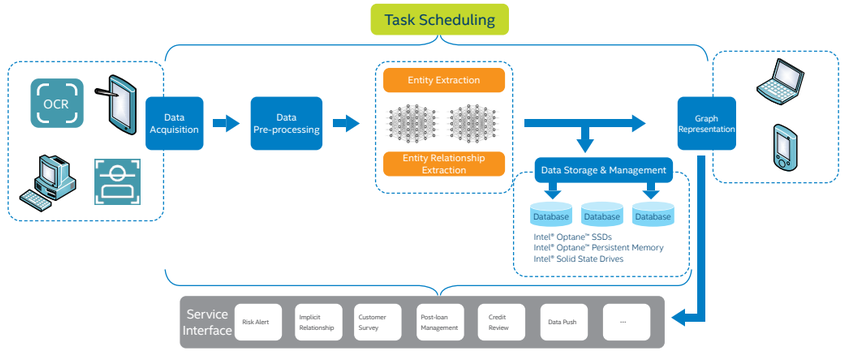


Figure 2-7-2 Components of a Common Knowledge Graph System in the Financial Industry

In the core entity & entity relationship extraction module, the system will first construct entities such as companies and individuals through AI algorithms; then, the entity attribute mining module and relationship mining module will analyze entity data using AI algorithm engine to obtain relationships and attributes between entities, and finally transform them into the knowledge graph data structures with entities, entity attributes, and entity relationships.

As shown in Figure 2-7-2, the representation module is typically able to visually display the entities, attributes, and relationships described above to users on various types of terminals. Generally, there are two display modes: the gird mode that displays the relationships with the central entity as well as relationships between other entities, and the tree model, in which the entities associated with the central entity are displayed.

The data extracted using relationships between entities can be stored in the database of financial institutions via data storage & management modules, and can also be invoked by internal and external systems in different scenarios through service interfaces; for example, they are available for information filtering and retrieval by using the internal Wiki system, or they can be used to provide unified external data services via the integrated plug-ins, such as displaying real-time information on enterprise websites.

Entity extraction method

As mentioned in the previous section, in a knowledge graph system built for the financial industry, the core module is to extract relationships between entities. BERT (Bidirectional Encoder Representations from Transformers) is a typical entity extraction model, which, as a bidirectional transformer encoder, essentially pre-trains the language representations, in other words obtaining a generic language comprehension model by training on a large corpus of texts.

The classical BERT structure is shown on the left side of Figure 2-7-3, which uses the Transformer Encoder model as the language model. The Transformer model structure is shown on the right side of Figure 2-7-3. This model discards the classical deep learning model structure such as RNN/CNN, and instead uses the Attention mechanism to compute the input-output relationship. Thanks to the Transformer Encoder, the Attention computing at each moment is possible to obtain the input of all moments. The advantage of the BERT model is that, a lot of grammar and semantic knowledge of natural language can be acquired through unsupervised learning, and better inference can be achieved by fine-tuning on small datasets.

For more information on the BERT model, see Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/pdf/1810.04805.pdf

Considering the structured data features commonly found in the financial industry and the optimizations provided by the Intel® architecture-based hardware and software products, users can adopt some optimized model variants, such as the BERT Word Vector + BiLSTM + CRF three-layer entity recognition model shown in Figure 2-7-4.

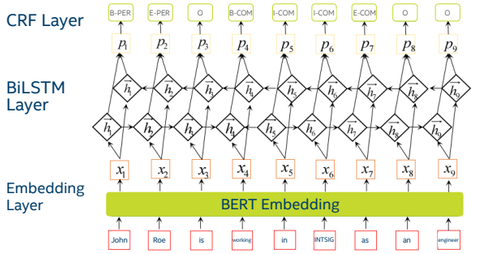


Figure 2-7-4 Diagram of BERT Word Vector + BiLSTM + CRF - a Three-layer Model Structure

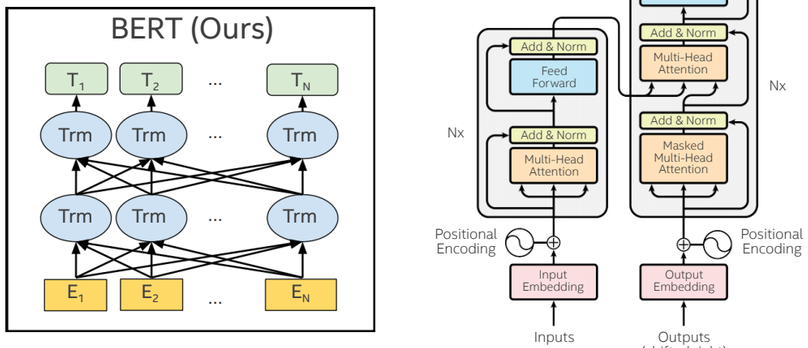


Figure 2-7-3 BERT Structure and Transformer Model[29]

[29] Related description on the Classical BERT model cited from Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding: https://arxiv.org/pdf/1810.04805.pdf

This model consists mainly of a three-layer network structure. The bottom layer is the Embedding Layer, which uses the BERT word vector to semantically represent each word, and the second layer is the BiLSTM Layer, which is a bi-directional LSTM model structure. As mentioned in the previous section, LSTM is an important variant of RNN, which can be designed with a special gate structure to avoid the long-term dependence problem and substantially improve the memory duration. The LSTM-based network structure is well suited for use in sequence-based analysis tasks. The bi-directional LSTM has both forward and backward propagation capabilities, which allows the network to better learn information about contextual semantics; the third layer is the CRF layer, which is trained to learn the transition probability matrix in the conditional random field, allowing for adding inter-label constraint relationship features.

Entity relationship extraction method

Entity relationship extraction, as an important task of information extraction, refers to the extraction of pre-defined entity relationships on the basis of re-identification of entities. As shown in Figure 2-7-5, the relationship between an entity pair can be described as a relationship triad <Entity M:Relationship X:Entity N>.

Figure 2-7-5 Architecture Diagram of Entity Relationship Extraction

For entity relationship extraction, common deep learning methods include TextCNN model, and Enhanced Representation from kNowledge IntEgration (ERNIE) model. The input to the TextCNN model is the word vectors in sequence, in other words the word vector will pass through the convolutional and pooling layers to get the final feature vector (here you can also set up the multi-layer convolution), and its last layer accesses the fully connected Softmax and outputs the category of entity relationship.

The ERNIE model is an improvement on the classical BERT model. Firstly, it pre-trains the model by masking semantic units such as words and entity concepts in Masked LM, so that the model's representation of semantic knowledge units is more realistic, and secondly, this model also introduces data corpus from more sources for training. When applied to entity relationship extraction, the ERNIR + Softmax classification model can be used to train on the paragraph input consisting of sentences where related entities are located, so as to predict the interrelationships between entities.

■ Application of Knowledge Graph in Financial Industry

Extensive use of knowledge graph in the financial industry

As the knowledge graph technology continues to evolve, its applications and market size are expanding at an accelerated pace. According to data from industry research and reports, the market size of intelligent big data covering the knowledge graph and the natural language processing applications was approx. RMB 10.66 billion in 2019, and this figure is expected to exceed RMB 30 billion by 2023. Of these, the financial industry now accounts for the largest share[30].

After decades of development and accumulation, the IT and other related systems in financial institutions often acquire and store massive amounts of structured business data. This data contains a large amount of multi-dimensional company and individual information, and is a treasure trove for the financial industry to drive operational efficiency, improve user experience, and reduce enterprise risk.

Using this data to facilitate the building of a knowledge graph system on efficient hardware infrastructure can help financial institutions directly extract entities, entity attributes, and relationships between entities in the data, and present the gridded chain of information to users in a concise and clear manner, effectively avoiding the performance shortcomings encountered in traditional relational databases when querying interactively.

For example, from data such as company registration data and business logs, you can extract entities such as company names and their directors as well as supervisors and management, or you can use business data such as shareholder relationships, external investments, capital flows and guarantees to construct relationships between entities. As shown in Figure 2-7-6, financial institutions are able to use specific big data analytics algorithms based on a large graph database to visualize the connections behind the curtain. This allows for more efficient avoiding of associated risk between different companies.

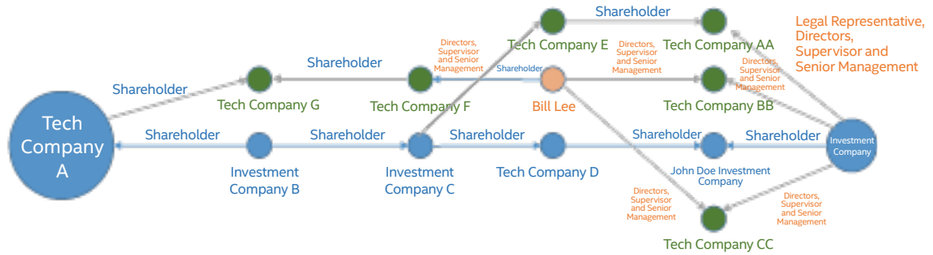


Figure 2-7-6 Mining Inter-entity Relationships using Graphical Relationships (Source: INTSIG Enterprise Graph)

[30] Data cited from iResearch 2020 China Knowledge Graph Industry Research Report: http://report.iresearch.cn/report_pdf.aspx?id=3553

Currently, there are the following typical application scenarios of knowledge graph in the financial industry:

- **Financial risk prevention and control with graph relationship mining and risk mining technologies**

Due to the intricacies of enterprise equity relationships, it is inevitable that commercial banks and other financial institutions may make errors or omissions when carrying out credit risk control, loan audit, credit expectation prevention and control. By building enterprise profiles on the big data platform and effectively storing and managing entities, relationships, events and attributes in the knowledge graph database, they can more effectively mitigate the above risks, and enable account managers, risk management department, credit approval department and other staff to better discover the risk relationships hidden under the complex network as well as the implicit relationships.

- **Building a financial risk prevention and control system with litigation relationship extraction**

The litigation relationship and the financial risk are closely related. As China's judicial data is gradually going into public domain, financial institutions can obtain more and more information on litigation case decisions through various means, but the judicial documents currently available on the Internet often consist of large paragraphs of text, which is not conducive to the retrieval, analysis and utilization of litigation cases. In this case, the knowledge graph can be used to quickly comb through various entities (including the case type, plaintiff, defendant, and amount of money involved) and entity relationships (plaintiff - responsible person, plaintiff - principal agent and other key relationships) appeared in various types of judicial documents, so that financial institutions can accurately determine the credit rating of relevant enterprises and individuals when carrying out line of credit approval, related-party guarantee and other businesses, and mitigate potential litigation risks in advance.

- **Implementing enterprise operation risk prevention and control with information available on the Internet**

Any unexpected or high-profile events may have an impact on enterprise operation and create an implicit operation risk. For example, an unexpected pest infestation in area A may reduce the yield of crop B, which in turn may reduce the sales of the specific agricultural machinery C, and ultimately affect the loan repayment capability of the agricultural machinery manufacturer D. Traditionally, this long chain analysis capability was performed by experienced experts, which has low accuracy and timeliness. Now, with the addition of NLP technology and knowledge graph system, financial enterprises can quickly collect and identify all kinds of subtle information available on the Internet and take relevant countermeasures.

Typical application scenarios in the financial industry: enterprise graph relationship mining and risk mining

Enterprises are the most important customer base of commercial banks and other financial institutions, and their production and operation situations have a great influence on the loan, credit and other relevant businesses of commercial banks. Modern enterprises often have very complex equity relationships, which are interlaced with each other, forming a related community; meanwhile, enterprise risk arises and develops during a series of activities such as production and operation. It is a dynamic process and is highly transmissible. In addition, risks may accumulate, amplify, or even break out through specific transmission mechanism, and may eventually lead to a cascading crisis. Therefore, financial institutions cannot understand the full picture of risk if they only know the production and operation of a single enterprise or a single sector. A growing number of financial institutions are now choosing knowledge graph to implement enterprise graph relationship mining and risk mining.

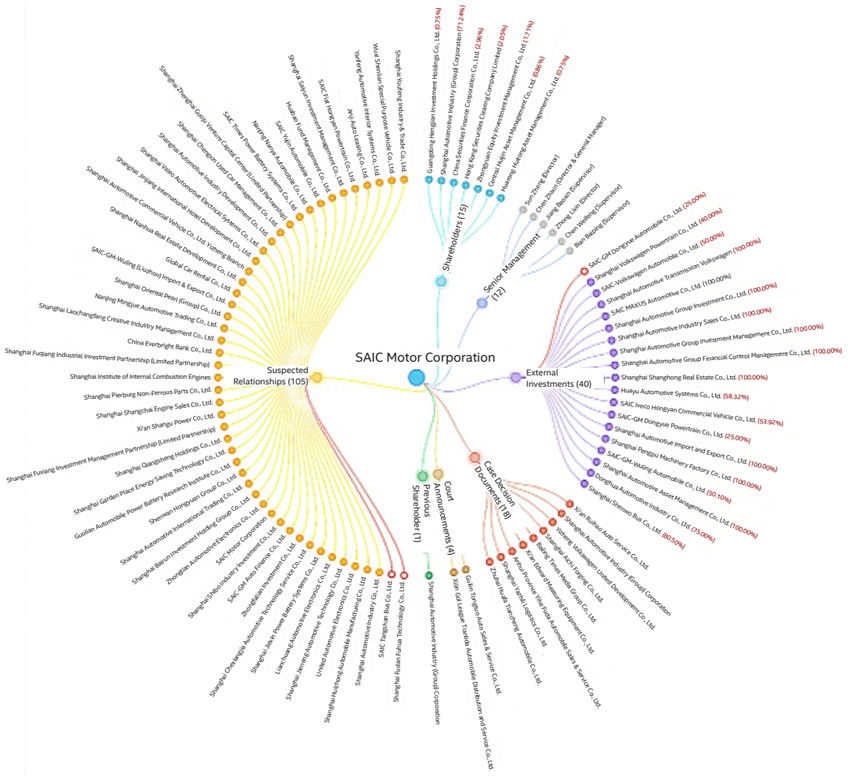


Figure 2-7-7 Example of Enterprise Graph (Source: INTSIG Qixinbao)

In general, enterprise risk can be classified in two ways:

- **Internal risk propagation:** Within a group, risks in some branches may transmit to each branch of the group or even the group as a whole.
- **External risk propagation:** Risks arising from external suppliers, distributors and partners of a group may transmit to the group as a whole or each branch of the group.

Therefore, financial institutions should first clearly understand the enterprise graph relationships by using the knowledge graph system, and then build an enterprise related risk propagation model to clearly show the risk propagation situation. As shown in Figure 2-7-7, the enterprise graph can show a full picture of the enterprise, which is a comprehensive illustration of the enterprise-related information, including shareholders, executives, external investments, court decisions, court announcements, previous shareholders and suspected relationships.

For example, when delivering loan, credit, and capital supervision services commercial banks are more concerned about the real decision makers and affiliates of an enterprise. To address these concerns, commercial banks traditionally need to perform manual investigation and maintenance. However, as the enterprise size is increasingly large, such work becomes time-consuming and labor-intensive, and each time an affiliate is added or a change in ownership occurs, it takes a long time for the relevant information to be transmitted to financial institutions. Now, on the one hand, with the help of the enterprise graph, financial institutions can trace the shareholders of an enterprise, view the capital contribution percentage of each shareholder, and obtain in-depth information about the actual controller; on the other hand, by viewing the capital contribution percentage and risks of related shareholders, financial institutions can also monitor the current operation of the enterprise, drill down into its external investments, display the investment path at multiple levels, analyze the shareholding relations, and uncover the cross-shareholding in investment relations.

After obtaining the enterprise graph, financial institutions can further build an enterprise associated risk propagation model based on their risk alert signal system, indicator database, or other abilities. For example, by using the graph technology, semi-supervised label propagation clustering, and neighboring network algorithms, they can gradually iterate on the data and model, facilitating the analysis of event + industry chain risk propagation model.

■ Challenges to the Use of Knowledge Graph in the Financial Industry

The evolving knowledge graph technology, while providing in-depth and high-value information mining capabilities for users in the financial industry, also puts forward higher requirements for relevant algorithms, computing power and data capabilities, which challenges traditional IT systems in financial enterprises, specifically:

- The increasing amount of information about entities and entity relationships in financial data poses a greater challenge to the computing, storage and transmission capabilities of the existing IT systems in financial institutions. Generally, when carrying out model iteration on high-dimensional structured financial data, the size of intermediate results often reaches the GB or even TB level, which requires a core processing platform with a higher clock frequency, more cores and threads, as well as greater memory capacity and performance.
- The knowledge graph system built by financial institutions adopts a range of technologies such as training & inference, high-performance storage and machine vision, requiring a complete, end-to-end technology chain that covers the deep learning technology to support the whole scenario.
- Building a highly accurate entity corpus requires a lot of investments in manpower and materials as well as a great deal of operation training for annotation personnel; Meanwhile, the fast-growing financial business data makes the entity corpus constantly change, resulting in inconsistency between the old and new corpus. Some large financial institutions may add tens of millions of data entries every day, so the financial industry needs to introduce new deep learning methods, especially the unsupervised deep learning methods, in building knowledge graph systems to cope with these changes.
- Also, the introduction of more deep learning methods requires the underlying hardware platforms to deliver new optimizations for the processor platforms and deep learning frameworks in response to the demands for new methods.

Intel's cloud-to-terminal full-stack platform includes a rich set of hardware and software products to address these challenges in knowledge graph applications, and the next section will detail how they support a variety of deep learning methods to solve these issues.

Advanced Hardware and Software Products Provide Support for the Deep Learning-based Knowledge Graph

As we can see, in order to achieve better performance, financial enterprise's knowledge graph system is required to adopt better computing and storage hardware as well as corresponding software optimization technologies in every portion of the system, including data acquisition/processing, knowledge graph construction/optimization, data storage/management and the final graph representation. To this end, as shown in Figure 2-7-8, Intel provides a holistic support to the knowledge graph system with its complete technology chain.

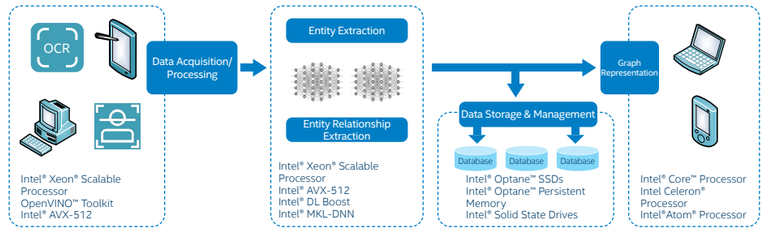


Figure 2-7-8 Intel Provides a Holistic Technical Support for Knowledge Graph System

■ OpenVINO™ Toolkit Helps Improve Data Acquisition Efficiency

In the data acquisition/processing phase, on the one hand, the new system needs to use various crawling technologies to acquire a large amount of public opinion information on the Internet. The high-speed crawler system often uses the large-scale parallel processing, which is where the manycore and high-frequency Intel® Xeon® Scalable Processor comes in. Also, the processor's integrated Intel® AVX-512 technology can facilitate the parallel processing of the crawler system. On the other hand, for large amounts of non-textualized information, Intel introduces several open source technologies such as the OpenVINO™ Toolkit to empower commonly used OCR recognition algorithms.

The OpenVINO™ Toolkit is an open source software toolkit from Intel, which is designed to accelerate image and video processing as well as deep learning inference and deployment. It not only delivers a number of built-in traditional APIs from the OpenCV and OpenXV vision libraries to accelerate and optimize image and video processing, but also includes a deep learning deployment toolkit to enable systems to fully leverage the computing power of Intel® processors and to accelerate the operational efficiency of deep learning models at the hardware instruction set level.

In a latency priority test on an FPN-SSD built on the ResNet101 model, the open-source ONNX Runtime engine was used with and without the OpenVINO™ Toolkit plug-in (see **https://github.com/microsoft/onnxruntime/** for details). As shown in Figure 2-7-9, the test results indicate that, when the number of OpenMP threads is 24, the ONNX Runtime engine with OpenVINO™ Toolkit achieved a 2.10x performance improvement.

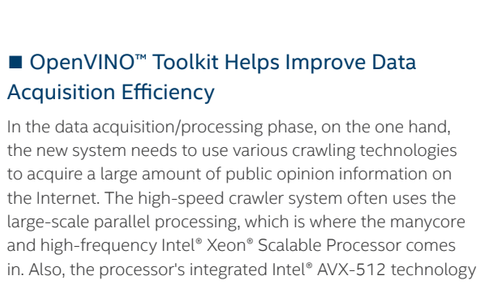


Figure 2-7-9 OpenVINO™ Toolkit Normalized Performance Comparison[31]

■ Intel® Processor Provides Powerful Computing Performance for Model Inference

Intel® Xeon® Scalable Processor can effectively speed up the inference of deep learning models such as entity/entity relationship extraction in knowledge graph systems during the knowledge graph building/optimization phase. The new generation Intel® Xeon® Scalable Processor not only integrates up to 56 processor cores and 112 threads, but also features a fully upgraded and optimized micro-architecture with faster and more efficient cache for improved processing performance, and supports up to 36 TB of system-level memory[32]. In addition, the Intel® AVX-512 instruction set integrated in the next generation Intel® Xeon® processor provides a wider vectoring capability, allowing users to design a system with more underlying floating-point computing optimizations.

The introduction of Vector Neural Network Instruction (VNNI) also enables the new generation Intel® Xeon® Scalable Processor to demonstrate a dramatic increase in deep learning inference speed, with up to 30 times higher inference performance compared to the previous generation[33], greatly improving the efficiency of knowledge graph system.

Meanwhile, Intel provides a rich library of optimized functions for each mainstream deep learning framework, such as TensorFlow and Caffe, and the introduction of these function libraries, for example Intel® MKL and Intel® MKL-DNN, can significantly improve the performance of the deep learning-based knowledge graph system when running on Intel® processor-based platforms.

For more information about Intel® MKL and Intel® MKL-DNN, please refer to relevant content in the Technologies section of this guide.

■ New Storage Technologies Provide Effective Data Support for System

The knowledge graph's capability to provide users with high-value information relies on its ability to train and infer on massive amounts of data by using deep learning methods. With the increasing size of data, especially in the financial industry, large financial institutions may add tens of millions of structured data entries every day, with data increments of over 100GB. In the course of training and inference on deep learning models, for example when extracting entity/entity relationships, the high dimensional nature of financial data may result in intermediate files in GBs or even in TBs.

In traditional storage architectures, such amount of data is mainly stored by hard disk drives (HDD) or solid state drives (SSD), but the real-time requirement of knowledge graph systems (especially in critical application scenarios such as financial risk control) demands for higher performance and lower latency. In this case, installing more DRAM memory will undoubtedly bring better performance, but it will also result in a dramatically higher cost to the user.

As shown in Figure 2-7-10, storage products at all levels built on Intel® Optane™ technology can meet different needs of knowledge graph systems for the above data storage. On the one hand, Intel® Optane™ Persistent Memory built on the 3D XPoint™ memory media offers similar read and write performance and access latency as DRAM memory, as well as better persistence than SSDs, providing comparable performance as DRAM memory in various high concurrency applications in the financial industry. By virtue of its high capacity, it also allows financial institutions to easily build in-memory databases with TB size. In addition, the non-volatile nature of the memory provides better system availability.

[31] Data from INTSIG's internal test; test configuration: processor: Intel® Xeon® Gold 6248R, 3.0GHz, 24 cores/48 threads, fixed input size: [1,3,768,768]; number of iterations: 50; test object: FPS

[32] Relevant parameters cited from the Intel website: https://ark.intel.com/content/www/us/en/ark/products/194146/intel-xeon-platinum-9282-processor-77m-cache-2-60-ghz.html

[33] Data cited from Intel website: https://www.intel.com/content/www/us/en/technology-provider/products-and-solutions/xeon-scalable-family/moving-ai-to-the-edge-article.html

For example, in the event of downtime or power failure, the system can be recovered in a fraction of the time by taking advantage of the non-volatile nature of the Intel® Optane™ Persistent Memory.

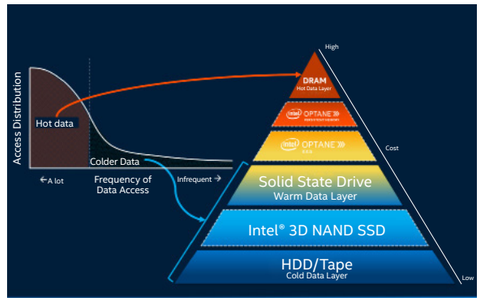


Figure 2-7-10 Performance and Cost Comparison of Various Types of Storage Hardware

Intel® Optane™ SSDs provide knowledge graph systems with efficient storage performance. Compared to Intel® Optane™ Persistent Memory, Intel® Optane™ SSDs offer a better balance of performance, capacity and cost. Utilizing a combination of advanced memory controllers, interface hardware and software technologies, Intel® Optane™ SSDs delivers low latency, high stability and more. Intel® Optane™ SSD DC P4800X, for example, offers up to 550,000 IOPS of random read/write and a read/write latency as low as 10 ms, making it ideal for multi-user, high concurrency applications in the financial industry. Its excellent Drive Writes Per Day (DWPD) also gives the system a longer life cycle, ensuring a better cost efficiency.

Use Case: Application of INTSIG Knowledge Graph in a Commercial Bank

Background

As one of the leading commercial banks in China, a commercial bank has been actively leveraging advanced IT technologies and AI capabilities to build its competitive advantage in IT technologies and e-banking platform, and to establish a highly reputable brand with product innovation and excellent customer service.

In recent years, as the bank continues to diversify its customer base, it has an increasing number of group customers with businesses across different industries and regions. These group customers, with varying creditworthiness, has increasingly complex relationships between each other, and it is not uncommon for banks to incur various risks or actual losses as a result of inadequate identification of affiliates of a group customer. In an internal group relationship validation, the bank checked the enterprise graphs of 10 groups, and found that 86 enterprises have erroneous data entries, and 92% of affiliates were not fully covered by the enterprise graphs, which obviously implies a significant amount of uncontrollable risk to the banking business.

In order to effectively control financial risks and ensure the safety of its credit assets, the bank hope to build a risk portal system based on knowledge graph technology to efficiently identify related enterprises and prevent business risks such as multiple credit, excessive credit and related guarantees. The bank has the following requirements for building the new system:

- The relationship graph can provide three types of external services: portal pages, relationship plug-ins and data interfaces, so that insider users can access the desired part of the content quickly and efficiently.
- Support for full search to show full information about an enterprise, ensuring that internal bank users do not need to switch between multiple systems and sites to increase operational efficiency.
- Quickly and easily view all types of relationships across all enterprises in the same portal, allowing for discovering information about suspected risks; Meanwhile, the new system needs to be integrated with the bank's existing risk information system to complement the bank's risk pre-analysis and risk early warning systems.
- Support for unified query of graph data, allowing internal bank users to fully understand customer information, and avoiding inconsistent information, credit duplication, incomplete information and other problems caused by isolated data. Also, the attribute and relationship mining capabilities of the knowledge graph can be used to uncover hidden information that can be used for credit review.
- Drill down into bank data, such as a large amount of high-value information hidden in transactions, guarantees and other data, enabling the bank to discover and eliminate risks early.
- Easy "out-of-the-box" deployment, visual presentation of relationships, and eliminating the need for extended development.

To satisfy the above requirements, INTSIG, which has been working on Knowledge Graph and NLP technologies for many years, worked with Intel, developed a new risk portal system based on Intel® Xeon® Scalable Processor, Intel® AVX-512, OpenVINO™ Toolkit and other Intel® architecture hardware and software products and technologies, and put it into operation smoothly. The new system has shown good performance in practice and has been well received by both internal and external users of the bank.

About INTSIG

Shanghai INTSIG Technology Co., Ltd., which has been focusing on the journey to business intelligence, is taking the lead in applying various AI deep learning technologies to traditional pattern recognition and enterprise knowledge graph system, by combining rich knowledge of customer risk management and marketing management, thereby allowing various users, especially those in the financial industry, to quickly develop a knowledge graph analysis and application capability covering internal and external data knowledge.

For more information, please visit the INTSIG website: https://www.intsig.com/

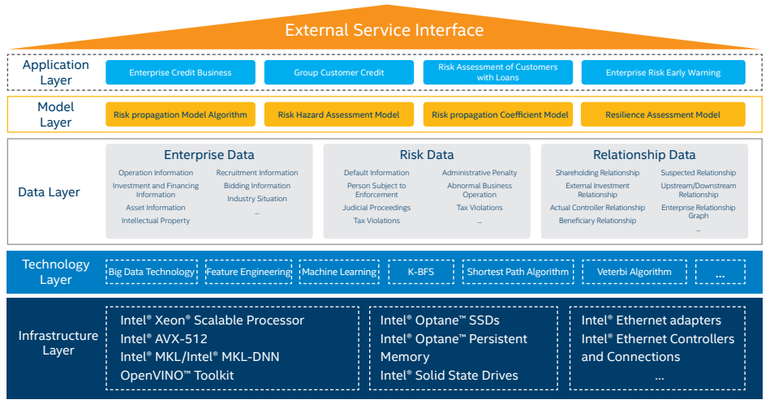


Figure 2-7-11 Risk Portal System Architecture of a Commercial Bank

INTSIG Knowledge Graph Solution Architecture and Optimization Solution

■ Overall Solution Architecture

As shown in Figure 2-7-11, the overall architecture of the new system consists of (from bottom to top) the infrastructure layer, technology layer, data layer, model layer, and application layer.

- At the infrastructure layer, a range of hardware and software products provided by Intel, including Intel® Xeon® Scalable Processor, Intel® AVX-512, Intel® MKL, and the OpenVINO™ Toolkit, provide powerful computing, storage, and transmission capabilities for the entire architecture.
- The technology layer consists of the deep learning and big data models such as BERT and TextCNN, which are introduced by INTSIG for higher-level applications and specifically optimized for Intel's hardware and software platforms.
- The data layer contains various types of internal and external multi-sourced data to which the system interfaces.
- At the model layer, INTSIG deploys risk propagation model algorithms and risk propagation coefficient models based on its knowledge of various business scenarios such as customer risk management and marketing management.
- At the top layer, the system can provide various types of risk control capabilities for the bank systems in the form of encapsulated APIs and integrated plug-ins.

It's worth mentioning that in the new project, based on the comprehensive full-dimensional data of 230 million enterprises in China, INTSIG built 10 customer relationship graphs for the user. As shown in Figure 2-7-12, these graphs include eight types of external data relationships: enterprise shareholding structure, external investment, actual controller, group relationship, suspected, litigation, address, and event relationships, as well as transaction and guarantee relationships that utilize the bank's internal transaction and guarantee data.

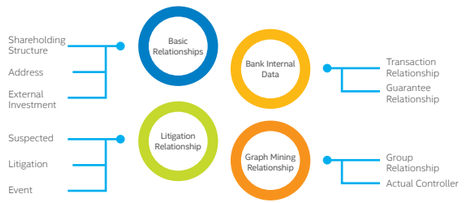


Figure 2-7-12 Customer Relationship Graph Construction

[34, 35] Data cited from INTSIG's internal test.

■ Optimization for the Entity Recognition Module based on the Classical BERT Model

In order to address the different data features involved in the new solution, INTSIG has also worked with Intel to develop a large number of specific optimizations based on the classical BERT model and the TextCNN model in the entity extraction and entity relationship extraction modules respectively.

Among them, the major optimizations for the entity extraction model include:

- The classical BERT model is fine-tuned to include a BiLSTM+CRF layer. The location or even direction information is necessary in sequence annotation task, but the classical BERT model reduces the location information, so we add the BiLSTM to better learn the dependency on the observed sequence; this optimization can improve the accuracy by about 0.5% (F1 Score) on the enterprise name entity that is common in the project (usually the length of this type of entity is long).
- Set a reasonable learning rate; the BERT and CRF layers are set with different learning rates, and the setting on the CRF layer is larger, which allows the CRF layer to learn quickly; the overall average accuracy (F1 Score) goes up by about 0.2% (F1 Score).
- Use one layer of BiLSTM. The BERT itself has a strong learning capability, and the underlying features are already rich enough, so that one layer of BiLSTM can achieve good results.

After implementing these optimizations, the model showed an overall accuracy (F1 Score) improvement of approximately 4%[34] compared to the baseline model (character-based BiLSTM+CRF model structure), and an overall average accuracy (F1 Score) improvement of approximately 0.8%[35] compared to using the BERT+CRF model alone.

■ Optimizations for the Entity Relationship Classification Module based on the TextCNN Model

The major optimizations include:

- Richer features are introduced into the Embedding layer for feature fusion, including character features, word features, lexical features, and more; the classification accuracy (F1 Score) is about 0.9% higher than using word features alone;
- Multiple different **kernel_sizes** are used to extract key information from text to better learn local relevance; in case of **kernel_sizes** = [3,4,5,6], the classification accuracy (F1 Score) improves by about 1%.
- Multiple convolutional layers are added, and the BatchNorm is merged into the convolutional layer, allowing for normalization and accelerating the fitting of training data by 0.3%.
- Because the distribution of categories is uneven, the data re-sampling and synonym substitution are performed to increase the diversity of the data; therefore, categories with a smaller amount improve by about 2%.

After implementing these optimizations, the overall accuracy (F1 Score) of the model improved by about 4.5% when compared with the baseline model (word-based TextCNN model structure, **kernel_size**=3, one convolutional layer, and no BatchNorm)[36].

Results

The risk portal project, with the knowledge graph technology at its core, went live at the commercial bank and received positive feedback from various departments within the bank. The system is continuously optimized through successive iterations, providing a continuously improving performance. Particularly, with the help of the knowledge graph enterprise database, big data mining and enterprise risk real-time monitoring technology, a branch of the bank has successfully and timely learned that a mortgaged equity has been frozen, which provides timely risk warning and eliminates the risk control problem caused by incomplete and unavailable information.

The bank's internal statistics show that, the system is used by 55 branches and subsidiaries; the total number of enterprises viewed is 75,596; the total number of visits is 91,423; and the number of key information actively pushed is 1,971; as shown in Figure 2-7-13, the enterprise basic information, risk information, and information of concern are the most popular modules among account managers, risk managers, and loan approvers respectively.

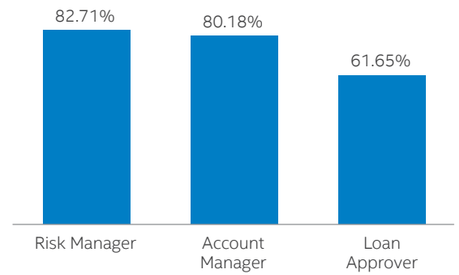


Figure 2-7-13 Coverage of New System Use by Different Users in a Commercial Bank

The system has also worked very well in public opinion prediction application. Data from real-world use shows that, the prediction time is only 0.232 seconds on the Intel® Xeon® Scalable Processor platform, which is significantly lower than the 0.5 seconds prediction time required by the bank. With a precision of 0.968 and a recall rate of 0.952, the F1 value reached 0.959, fully meeting the bank's expected target of F1=0.95[37].

Recommended Hardware and Software Configuration

The above financial system knowledge graph solution can be built with reference to the following Intel® architecture-based platform, with the following environment configuration:

Name	Specification
Processor	Intel® Xeon® Gold 6248R Processor or higher
Base Frequency	3.00GHz (Turbo Boost @ 4.00GHz)
Cores/Threads	24/48
HT	On
Turbo	On
Memory	384G (32G DDR4 2933MHz x12)
Storage	Intel® SSD D5 P4320 Series and above

Software Configuration

Name	Specification
Operating System	CentOS 7.7.1908
Linux Kernel	3.10.0-1062.1.2.el7.x86_64
Workloads	ResNet101 + FPN/ BERT NER
Compiler	gcc 7.5.0
Library	LLVM OpenMP runtime: Libomp-dev 5.0.1-1
OpenVINO™ Toolkit	OpenVINO 2020 R3

Conclusion

INTSIG is working with Intel to explore the high value information contained in massive amounts of business data by using knowledge graphs. The knowledge graph system built with the deep learning method enables users in the financial industry not only to effectively build a firewall against financial risks such as loan risk control, credit risk control and enterprise operation risk prevention & control by relying on capabilities such as graph relationship mining, risk mining and judicial relationship extraction, but also to quickly identify the impact of unexpected or high-profile events on enterprise operation from the Internet information and take corresponding measures.

Intel not only delivers the high-performance Intel® Xeon® processor platform as the computing core for the new deep learning-based knowledge graph system, but also provides a range of hardware and software technologies such as Intel® AVX-512, Intel® MKL and the OpenVINO™ Toolkit to optimize the system's performance. Now, the new knowledge graph system has been deployed in a commercial bank, bringing users convenient and efficient financial risk control capabilities, while winning unanimous praise from users in the bank. In the future, INTSIG also plans to work with Intel to further explore the application of knowledge graph in various industries. INTSIG hopes to further expand Intel's AI innovation ecosystem by relying on Intel's cloud-to-terminal technology advantages and full-stack platform to promote AI development and breakthroughs in various industries and scenarios, accelerate the implementation of intelligent applications, and facilitate the intelligent transformation of various industries.

[36] Data cited from INTSIG's internal test.

[37] The test configuration: processor: Intel® Xeon® Gold 6248R processor, 3.00GHz, 24 cores/48 threads; the test scenario is a random selection of 100 news articles with a total of 1,327 sentences, Batch Size is set to 6, and the test results are the average prediction time and overall accuracy of a single batch.

End-to-end Unified Big Data and AI Platform Facilitates a Seamless Transition from Big Data to Deep Learning in the Financial Industry

Explore the Deep Learning Methods that are Built on Financial Big Data

Deep Learning Methods Provide AI-powered Capability for More Financial Services

■ Big data has become the infrastructure of the financial industry

The continuous integration of the financial industry with the mobile Internet, big data, 5G, cloud computing and other emerging technologies is giving rise to new financial business models such as online banking and online payment, as well as driving financial institutions to transform to an intelligent and smart IT environment. In this regard, like the "Seven Connections and One Leveling" project that needs to be carried out at the beginning of urban construction, the building of big data platform and the higher-level applications, is also a necessary "infrastructure" project before financial institutions provide new services and new functions.

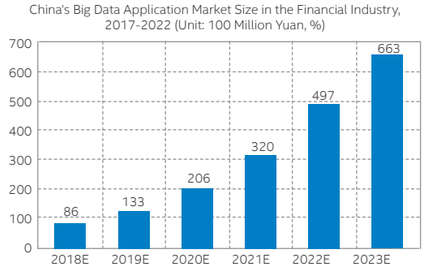


Figure 2-8-1 China's Big Data Market in the Financial Industry is Increasing

As shown in Figure 2-8-1, one figure shows that the market size of big data in China's financial industry is expanding at a phenomenal rate. In addition, a number of reports have pointed out that the financial big data platforms are actively changing in the face of innovative business scenarios, and that at present they typically need to have the following characteristics:

- **Good real-time computing support:** More complex business scenarios and stricter regulatory requirements, such as real-time risk control, transaction alerts and anti-fraud, require higher On-Line Transaction Processing (OLTP) performance for big data platforms.The infrastructure built with Intel® architecture hardware and software products, as well as efficient computing and storage engines, are enabling big data platforms to provide real-time computing and storage capabilities in the millisecond and even microsecond range.
- **Adopting cloud platform:** More financial institutions are deploying their big data services on different types of cloud platforms (public, private and hybrid clouds) based on their business needs to build a multi-dimensional data security system and off-site disaster recovery solution. Cloud-based big data services also facilitate on-demand delivery and single sign on.
- **Distributed architecture and data lake:** The new generation of big data platforms often choose distributed system architectures such as Hadoop + Apache Spark. The x86 server cluster-based big data platforms provide good horizontal scalability, linear storage and computing resources, which can significantly reduce computing and I/O resource bottlenecks. In order to move forward quickly, some financial institutions have even begun to build the data lake solution with hybrid architecture to empower the On-Line Analytical Processing (OLAP).

The above trends clearly show that the integration with AI capabilities is undoubtedly becoming a very interesting and increasingly important effort in the financial industry. As we all know, the development of AI is supported by algorithms, computing power and data, and big data in the financial industry can act as underlying driving force for AI-based financial applications, providing rich training datasets for a range of deep learning/machine learning methods. In recent years, many financial institutions have developed diverse AI applications in areas such as the anti-fraud, financial risk control, credit risk prediction and intelligent customer service based on the financial service big data and the infrastructure built by Intel® architecture software and hardware products. For detailed analysis of these solutions, please refer to previous sections.

■ Building Financial Big Data Platform and AI Capabilities

To store massive financial service data, many financial institutions have built their big data platforms on Hadoop/Spark distributed system storage architecture, and use them to develop various big data applications or build AI capabilities.

A typical big data platform in financial institutions is shown in Figure 2-8-2. The underlying data source consists of structured and unstructured data such as customer data, transaction data and account data. On top of that is the data service capability built by a cluster of Hadoop servers, providing the platform with distributed file system, resource/task scheduling, and computing services. The highest layer includes various data applications that are developed by financial institutions for their needs.

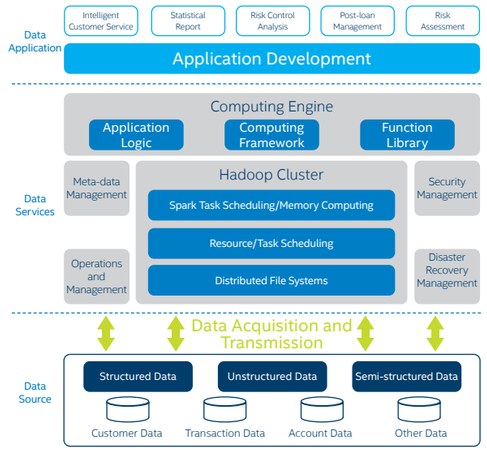


Figure 2-8-2 A Typical Big Data Platform in Financial Institutions

At the core of the platform, the Hadoop cluster provides efficient distributed file systems, computing frameworks, data warehouses, and scheduling capabilities for massive amounts of financial service data. In recent years, advocated by companies like Intel and Cloudera, more and more financial institutions are adopting Spark as a computing engine for their big data platforms. Compared to MapReduce in Hadoop, Spark enables quasi-real-time streaming processing for uninterrupted input data from different sources, as well as deep mining and analysis of massive amounts of data using a large-scale, complex machine learning/deep learning and graph computing. In addition, Spark can use a distributed high-speed memory cache to support interactive, iterative computing and data analysis, allowing data computing speed to increase dramatically.

Based on the above big data platform, many financial institutions have explored and implemented a large number of AI applications. For example, some commercial banks are using the XGBoost algorithm to build loan risk prediction models, and are achieving good results. However, it is worth noting that financial institutions are still mainly using machine learning methods for AI applications in intermediate services such as the third-party collection and payment, settlement and credit card issuance. The reason is that the existing user and capital data in commercial banks are generally wide table, structured data, and have significant serialization characteristics, so it is easier to explore AI applications with machine learning methods.

As the scale of AI applications continues to increase, the use of traditional machine learning methods alone is showing their shortcomings. For example, in a typical capital flow prediction scenario, when the scale and relevance of a capital flow reaches a threshold, it will exhibit high intelligence, strong correlation, tight coupling and randomness, making it a complex non-linear power system, and deep learning methods have better performance in such prediction scenario.

In addition, traditional machine learning methods are also difficult to perform automatic learning, unable to self-learn from complex data, and unable to optimize AI performance through iteration. Meanwhile, the massive amount of data owned by big data platform in financial institutions can also provide the large datasets needed for deep learning training. And all of these provide the necessary prerequisites for financial institutions to carry out the exploration of deep learning methods by using financial big data.

■ Challenges in Building Deep Learning Methods on Existing Big Data Platform

In financial institutions, transforming from big data platforms to deep learning methods can't be achieved overnight. In this process, data specialists and AI application engineers in financial institutions may encounter the following issues:

- The existing Hadoop/Spark distributed system storage architecture, while providing powerful storage capabilities, also deepens the complexity of deep learning frameworks for acquiring and leveraging data. If there is a lack of end-to-end available platforms, building deep learning models on existing big data platforms will indirectly raise the bar for financial institutions to carry out research and exploration of deep learning methods.
- For different business scenarios, financial institutions will choose different deep learning frameworks and supporting software and hardware infrastructure, including data platforms and computing platforms, according to their own circumstances when building AI models, which will affect the prediction efficiency and accuracy and cost a lot of debugging overheads.
- Existing big data platforms often lack a unified hardware and software integration system, and cannot effectively provide optimization and acceleration for the underlying computing power when using different deep learning frameworks. Also, the fine-tuning of deep learning framework and their codes will consume a lot of manpower and time.

In order to help more financial institutions efficiently combine their existing big data platforms with deep learning exploration, Intel is providing a range of hardware and software products and technology solutions for big data platforms and deep learning methods, and deeply collaborating with financial partners to provide an end-to-end platform of methods, processes, and tools (e.g. Analytics Zoo), allowing for "seamless switch" from big data to deep learning. At present, this approach has been used in several real-world deployments, and achieved satisfying results.

Explore the Deep Learning Methods that are Built on Financial Big Data

■ Building Deep Learning Methods for Capital Flow Prediction Scenario

To switch from machine learning to deep learning on an existing financial big data platform, financial institutions need to develop a proven set of methodology and switch process in a number of implementations by addressing the following issues:

- How to select appropriate deep learning methods and algorithms based on the business scenarios of financial institutions.
- How new deep learning methods and algorithms interface with the existing big data platform to enable effective data interaction between the big data platform and the deep learning system.
- How to use the existing computing power and resources of the financial institution's IT system to select a reasonable optimization library that provides acceleration for deep learning methods.
- How to reduce the cost burden on financial institutions caused by the required hyper-parameter tuning in the deep learning methods.

Thus, as shown in Figure 2-8-3, to help financial institutions explore deep learning methods on the big data platform, Intel needs to work with users to build an end-to-end platform of methods, processes, and tools.

First, more and more deep learning methods have been applied to various business scenarios in the financial industry, for example, the advantages of deep learning methods in image analysis are utilized to develop AI applications such as face detection & recognition and image segmentation, and then used in insurance claims, and intelligent customer service. However, due to the fact that traditional deep learning methods are not advantageous in processing discrete data like structured financial data, machine learning methods are still predominantly used in scenarios such as capital flow prediction. In addition, the discrete data requires large amounts of labeled data for training, complex theoretical analysis, and tremendous parameter adjustment efforts.

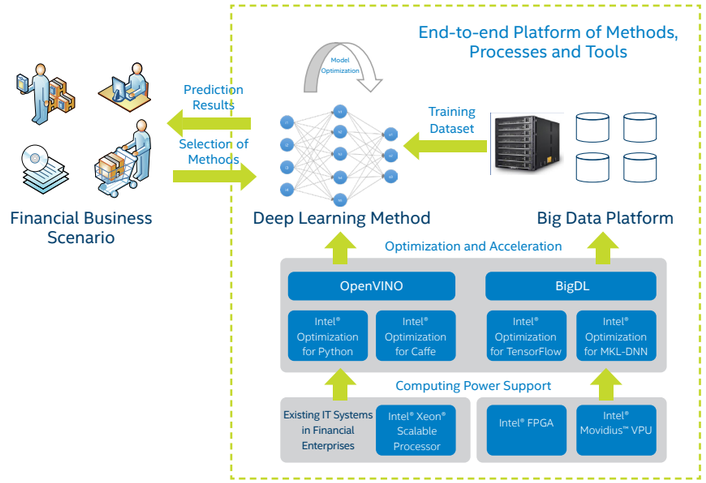


Figure 2-8-3 Building Deep Learning Methods on the Big Data Platform Requires an End-to-end Platform of Methods, Processes and Tools

As deep learning algorithms, computing power and related optimization and acceleration capabilities continue to improve, and the increasingly sophisticated financial big data platform can provide more high-quality training datasets for deep learning methods, more and more financial institutions are exploring the application of deep learning methods in capital flow prediction and so on. These methods include Multilayer Perceptron (MLP), Autoregressive Integrated Moving Average Model (ARIMA), DNN, CNN, and LSTM. Financial institutions can choose the most appropriate algorithmic model according to their applications and data types. In the following section, we will briefly describe the MLP model that is widely adopted in the capital flow prediction solution.

Secondly, the financial big data platforms are often deployed in a distributed architecture, therefore, the deep learning methods also need to be deployed in a proper manner. Some financial institutions are trying to implement the BigDL, one of the distributed deep learning frameworks, which is a distributed deep learning library open-sourced by Intel and built on the Spark computing engine. With the introduction of BigDL, users can run deep learning applications such as capital flow prediction as standard Spark programs directly on Hadoop/Spark cluster of existing big data platform.

The building of deep learning methods also requires a large amount of computing power resources, which needs to not only make use of resources available in the user's existing IT systems (in other words rich general-purpose processor computing power) to avoid repeated investment and reduce the user's CAPEX cost, but also carry out processor-specific optimizations to improve the efficiency of computing power. For example, when an AI solution adopts the TensorFlow framework, users can introduce the Intel® Optimizations for TensorFlow. The optimized framework can optimize both the graph process using the Intel® MKL library and multiple threaded libraries, thus improving the utilization of computing resources and accelerating the execution of deep learning methods.

Finally, traditional machine learning/deep learning methods require to label the training data and tune the hyper-parameters, which consumes a lot of manpower. Therefore, when exploring deep learning methods for future applications, users in the financial industry are also eager for new solutions with the ability to automatically tune deep learning methods.

■ Deep Learning Methods based on the MLP model

The MLP model is widely used in capital flow prediction solutions because of its simplicity, efficiency and ease of deployment. A typical MLP model architecture is shown in Figure 2-8-4, in which the neural network consists of fully connected layers and contains at least one hidden layer (for example, in Figure 2-8-4, there is only one hidden layer), the output of each hidden layer is transformed by an activation function, and the hyper-parameters of the MLP model are the number of network layers and the number of hidden units in each hidden layer.

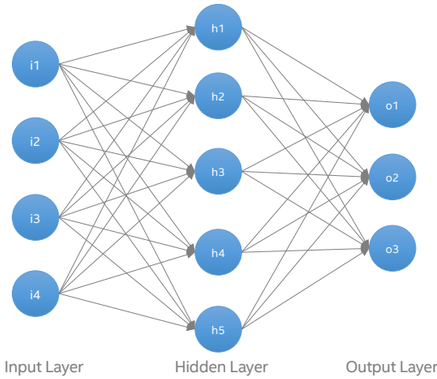


Figure 2-8-4 Typical MLP Model Architecture

The activation functions commonly used in the MLP model include the ReLU function, the Sigmoid function, and the tanh function, in which:

- The ReLU function provides a simple nonlinear transformation defined as ReLU(x) = Max (x, 0) for the given element x. Thus, the ReLU function retains only positive elements and clears out negative elements.
- The sigmoid function transforms the value of an element to between 0 and 1, i.e., sigmoid(x) = 1/(1+exp(-x)), and according to the chain rule, the derivative of the sigmoid function is sigmoid′(x) = sigmoid(x)(1−sigmoid(x)).
- The tanh function can transform the value of an element to between -1 and 1, i.e., tanh(x) = (1−exp(−2x))/(1+exp(−2x)). According to the chain rule, the derivative of the tanh function is tanh′(x) = 1−tanh^2(x), which reaches a maximum value of 1 when the input is 0; and the more the input deviates from 0, the closer the derivative of the tanh function reaches to 0.

Compared with other deep learning methods, the MLP model, on the one hand, has good nonlinear global effects and fault tolerance, and has an associative memory function, and in the case of a large training dataset, it can obtain excellent prediction results; on the other hand, the model also has a good parallel processing capability, which is what the manycore and high-frequency Intel®-based processors do best. Financial institutions can leverage the massive computing power of Intel® processors in their existing IT systems to improve the performance of AI applications.

In addition to the MLP model, Intel also helps financial institutions explore AI applications such as prediction and recommendation in different financial business scenarios using other deep learning models such as DNN, CNN and LSTM.

■ Intel Provides an End-to-end Unified Tool Platform for Exploring Deep Learning Methods

In addition to the selection of methods and processes, financial institutions need a unified, end-to-end tool platform to host these methods and processes when exploring deep learning methods on their big data platforms. Analytics Zoo, an open-source Big Data Analytics + AI platform introduced by Intel, on the one hand, seamlessly integrates Spark, PyTorch, TensorFlow, Keras and other software and frameworks into a unified system, and easily scales to the Hadoop/Spark cluster; on the other hand, it also incorporates multiple software libraries, such as Intel® MKL and Intel® MKL-DNN, to fully unleash the vector and deep learning instructions integrated into computing platforms such as the 2nd Generation Intel® Xeon® Scalable Processor, significantly increasing the speed of training and inference for AI applications.

With internally integrated modules such as Spark and BigDL, Analytics Zoo is able to seamlessly integrate with financial institution's existing big data platform architecture. As shown in Figure 2-8-5, on the one hand, Analytics Zoo allows users to seamlessly integrate the software and frameworks required by big data platforms and deep learning methods (such as Spark and PyTorch) in the capital flow prediction solution into the same process, which enables users to integrate the pipeline of data storage, data processing, training and inference into a unified infrastructure, thus significantly improving deployment efficiency, resource utilization and scalability, and reducing hardware management and system operation and maintenance costs.

On the other hand, Analytics Zoo also provides a large number of diverse, optimized function libraries for different computing processor platforms, such as Intel® MKL and Intel® MKL-DNN as well as optimized processor platform-specific deep learning frameworks and programming languages,

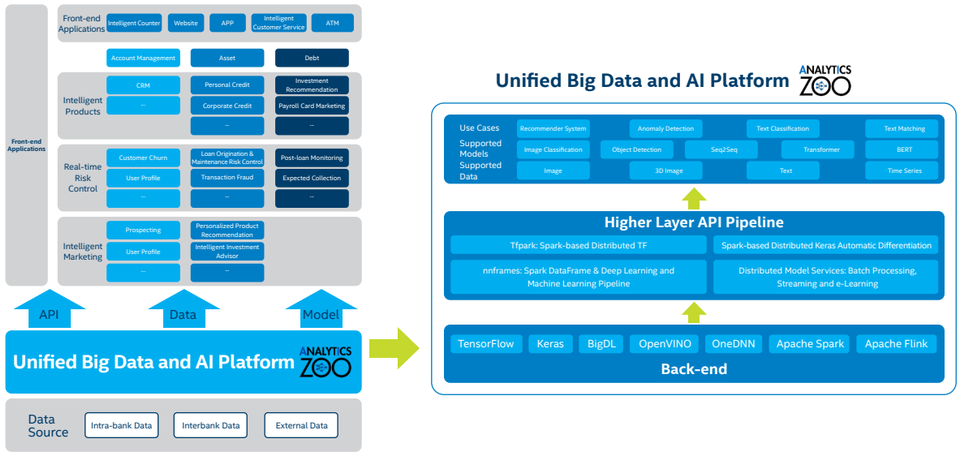


Figure 2-8-5 Analytics Zoo Integrated with the Financial Big Data Platform

like Intel® Optimization for TensorFlow, Intel® Optimizations for Caffe and Intel® Optimization for Python. In addition, Analytics Zoo comes with a rich set of pre-built functional components for different AI scenarios. For example, for capital flow prediction scenarios, Analytics Zoo can provide:

- **Common deep learning models:** MLP, DNN, CNN, LSTM, MTNet, and ARIMA;
- **Commonly used data pre-processing and feature engineering components:** Datetime features, Time diff, Log-transform, and Rolling window;
- **Common anomaly detection methods:** Percentile, Distribution-based, Uncertainty-based, and Autoencoder.

Finally, in the latest version of Analytics Zoo, the AutoML framework has been added to automate feature selection, model selection, and hyper-parameter tuning to further improve the productivity of prediction model.

■ Future-oriented AutoML Framework

Generally, data pre-processing and model optimization in the traditional deep learning or machine learning models need to be carried out by experienced data scientists, which undoubtedly increases the system maintenance pressure and labor costs for financial institutions. To this end, Intel has added the AutoML framework built on the open-source Ray Distributed Framework, to its latest Analytics Zoo Big Data Analytics + AI platform, automating processes such as feature generation, model selection and hyper-parameter tuning to further improve prediction efficiency.

As shown in Figure 2-8-6, the AutoML framework mainly consists of components such as FeatureTransformer, Model, SearchEngine and Pipeline, in which:

- FeatureTransformer defines the feature engineering process, including operations such as feature generation, feature scaling and feature selection.
- Model defines the model and the optimization algorithm used.
- SearchEngine is used to search for the best combination of hyper-parameters of FeatureTransformer and Model and is central to the control of the model training process.
- Pipeline is configured with the best end-to-end data analysis pipeline for FeatureTransformer and Model, and can be loaded and used repeatedly.

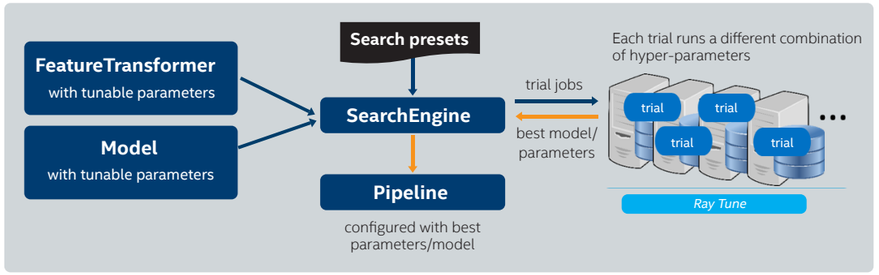


Figure 2-8-6 Typical AutoML Execution Process

Take the capital flow prediction in the financial industry as an example. First, AutoML sends the parameterized FeatureTransformer and Model to SearchEngine for instantiation. The SearchEngine then uses Ray Tune to run multiple rounds of trial jobs in the right cluster, each with a different hyper-parameter combination for feature engineering and model training. Finally, the system will select the best combination of hyper-parameters and model, and then send them to Pipeline for subsequent prediction training and inference.

For more AutoML codes, Demo and documents, see:

- Branch in Analytics Zoo Repo: https://github.com/intel-analytics/analytics-zoo/tree/automl
- AutoML Readme document: https://github.com/intel-analytics/analytics-zoo/blob/automl/pyzoo/zoo/automl/README.md
- Demo manual: https://github.com/intel-analytics/analytics-zoo/blob/automl/apps/automl/nyc_taxi_dataset.ipynb

■ Optimization Sample

Intel provides a diverse set of code optimization samples for users in the financial industry, enabling them to use Analytics Zoo more quickly and easily, explore deep learning methods on the financial big data platform, and get good results. The following is a code optimization sample for reference, which is divided into several main functional blocks:

- Define **HADOOP_CONF_DIR** and initialize **yarn**.
- Pre-process data, including data cleansing, data deduplication, and data splitting.
- Define various hyper-parameters of the model, preparing for subsequent model training and prediction.
- Prepare the model, including model definition, acquisition of feature sets, and preparation of cross validation datasets.
- Use Analytics Zoo for training.

Define **HADOOP_CONF_DIR** and initialize **yarn**:

```
from zoo.common.nncontext import *

if os.environ.get('PYTHONHOME') is not None:
    sc = init_spark_on_local(cores=1, conf={"spark.driver.memory": "20g"})
else:
    hadoop_conf_dir = os.environ.get('HADOOP_CONF_DIR')
    num_executors = 2
    num_cores_per_executor = 1
    os.environ['ZOO_MKL_NUMTHREADS'] = str(num_cores_per_executor)
    os.environ['OMP_NUM_THREADS'] = str(num_cores_per_executor)
    sc = init_spark_on_yarn(
      hadoop_conf=hadoop_conf_dir,
      conda_name=os.environ["ZOO_CONDA_NAME"],  # conda 环境名称
      num_executor=num_executors,
      executor_cores=num_cores_per_executor,
      executor_memory="20g",
      driver_memory="20g",
      driver_cores=1,
      spark_conf={"spark.rpc.message.maxSize": "1024",
            "spark.task.maxFailures": "1",
            "spark.scheduler.minRegisteredResourcesRatio": "1",
            "spark.scheduler.maxRegisteredResourcesWaitingTime": "100s",
            "spark.driver.extraJavaOptions": "-Dbigdl.failure.retryTimes=1"})
```

Pre-process data:

```
from pyspark.sql import functions as F
from scipy.stats import mode

#  数据集预处理
import pandas as pd
data=pd.read_csv("gen_data.csv").set_index('Cust_Id')
data=data.fillna(0)

for i in data.columns:
  if data[i].dtype != 'float64':
    print(i,data[i].dtype)

#  去除非特征列
fea_notF  = ['df_dt','core_cust_lev_cd','indus_cd','Rating','City_Cd']
data.drop(data[fea_notF],axis=1, inplace=True)


#  去除同值特征列
equi_fea = []
for i in data.columns:
  try:
    mode_value = mode(data[i][data[i].notnull()])[0][0]
    mode_rate = mode(data[i][data[i].notnull()])[1][0] / data.shape[0]
    if mode_rate > 0.6:
      equi_fea.append([i, mode_value, mode_rate])
  except Exception as e:
    print(i, e)
e = pd.DataFrame(equi_fea, columns=['col_name', 'mode_value', 'mode_rate'])
e.sort_values(by='mode_rate')

same_val_fea_to_drop = list(e.col_name.values)

for i in same_val_fea_to_drop:
  if i == 'if_lc':
    print(i)
    same_val_fea_to_drop.remove('if_lc')

data.drop(same_val_fea_to_drop, axis=1, inplace=True)

data[['if_lc']]=data[['if_lc']].astype('double')

from sklearn.preprocessing import StandardScaler

data_feature = data[data.columns[1:]]
data_label = data[data.columns[0]]

#  拆分数据集为训练集与验证集
from sklearn.model_selection import train_test_split
train_X,test_X,train_y,y_test = train_test_split(data_feature,data_label,test_size=0.3)
```

Define various hyper-parameters of the model:

```
epochs = 30
batch_size = 1024
classes = 1
learning_rate = 0.01

scaler = StandardScaler()
train_X = scaler.fit_transform(train_X)
test_X = scaler.transform(test_X)
```

Prepare the model:

```
import torch
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
from sklearn.metrics import roc_auc_score
from torch.autograd import Variable
from sklearn.model_selection import KFold
import numpy as np

# 定义模型
class MLP(torch.nn.Module):
    def __init__(self, n_feature, n_hidden, n_output, dropout=0.5):
        super(MLP, self).__init__()
        self.dropout = torch.nn.Dropout(dropout)
        self.hidden_1 = torch.nn.Linear(n_feature, n_hidden)
        self.bn1 = torch.nn.BatchNorm1d(n_hidden)
        self.hidden_2 = torch.nn.Linear(n_hidden, n_hidden//2)
        self.bn2 = torch.nn.BatchNorm1d(n_hidden//2)
        self.hidden_3 = torch.nn.Linear(n_hidden//2, n_hidden//4)
        self.bn3 = torch.nn.BatchNorm1d(n_hidden//4)
        self.hidden_4 = torch.nn.Linear(n_hidden // 4, n_hidden // 8)
        self.bn4 = torch.nn.BatchNorm1d(n_hidden // 8)
        self.out = torch.nn.Linear(n_hidden//8, n_output)

    def forward(self, x):
        x = F.relu(self.hidden_1(x))
        x = self.dropout(self.bn1(x))
        x = F.relu(self.hidden_2(x))
        x = self.dropout(self.bn2(x))
        x = F.relu(self.hidden_3(x))
        x = self.dropout(self.bn3(x))
        x = F.relu(self.hidden_4(x))
        x = self.dropout(self.bn4(x))
        x = self.out(x)
        return x

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# 将训练集分割为 5 折
folds = KFold(n_splits=5, shuffle=True, random_state=2019)
NN_predictions = np.zeros((test_X.shape[0], ))
oof_preds = np.zeros((train_X.shape[0], ))
x_test = np.array(test_X)
x_test = torch.tensor(x_test, dtype=torch.float)
test = TensorDataset(x_test)
test_loader = DataLoader(test, batch_size=batch_size, shuffle=False)
avg_losses_f = []
avg_val_losses_f = []

from bigdl.util.common import Sample

# 获取 FeatureSet
def get_featureset(x, y, shuffle=True):
    x = np.split(x.data.numpy(), x.shape[0])
    y = np.split(y.data.numpy(), y.shape[0])
    print(x[0].shape)
    print(y[0].shape)
    samples = [Sample.from_ndarray(np.squeeze(x[i]), np.squeeze(y[i])) for i in range(len(x))]
    sample_rdd = sc.parallelize(samples)
    return FeatureSet.sample_rdd(sample_rdd, shuffle=shuffle)

# 模型训练
for fold_, (trn_, val_) in enumerate(folds.split(train_X)):
    print("fold {}".format(fold_ + 1))
    x_train = Variable(torch.Tensor(train_X[trn_.astype(int)]))
    y_train = Variable(torch.Tensor(train_y[trn_.astype(int), np.newaxis]))

    x_valid = Variable(torch.Tensor(train_X[val_.astype(int)]))
    y_valid = Variable(torch.Tensor(train_y[val_.astype(int), np.newaxis]))

    model = MLP(x_train.shape[1], 512, classes, dropout=0.4)
    print(x_train.shape[1])
    loss_fn = torch.nn.BCEWithLogitsLoss()
```

Use Analytics Zoo for training:

```
from zoo.pipeline.api.keras.optimizers import AdamWeightDecay
    from zoo.pipeline.api.torch import TorchModel, TorchLoss
    from zoo.feature.common import FeatureSet
    from zoo.pipeline.estimator import *

    zooOptimizer = AdamWeightDecay(lr=learning_rate, weight_decay=1e-5)
    zooModel = TorchModel.from_pytorch(model.cpu())
    zooLoss = TorchLoss.from_pytorch(loss_fn)
    train_featureSet = get_featureset(x_train, y_train, shuffle=True)
    val_featureSet = get_featureset(x_valid, y_valid, shuffle=False)

    estimator = Estimator(zooModel, optim_methods=zooOptimizer)

    from bigdl.optim.optimizer import MaxEpoch, EveryEpoch
    from zoo.pipeline.api.keras.metrics import AUC
    estimator.train(train_featureSet, zooLoss, end_trigger=MaxEpoch(epochs),
                                checkpoint_trigger=EveryEpoch(),
                                validation_set=val_featureSet,
                                validation_method=[AUC()], batch_size=batch_size)
    torchModel = zooModel.to_pytorch()

# 预测
test_preds_fold = np.zeros((len(test_X)))
for i, (x_batch,) in enumerate(test_loader):
    y_pred = torchModel (x_batch).detach()
    test_preds_fold[i * batch_size:(i + 1) * batch_size] = sigmoid(y_pred.cpu().numpy())[:, 0]
    NN_predictions += test_preds_fold / folds.n_splits
for i in oof_preds:
  if i>0.5:
    print()

threshold = 0.5
result = []
for pred in NN_predictions:
    result.append(1 if pred > threshold else 0)

threshold = 0.5
result = []
for pred in oof_preds:
    result.append(1 if pred > threshold else 0)

# 计算 acc
from sklearn.metrics import accuracy_score
acc = accuracy_score(train_y, result)
```

Use Case: A Commercial Bank

Background

■ Big data has become the infrastructure of the financial industry

The payroll service is one of the important businesses of commercial banks, which is commissioned by enterprises and other employers to transfer the required payments such as labor remuneration to the designated employee accounts. This business can bring following benefits for commercial banks:

- The demand deposits in corporate accounts and individual payroll accounts have been a source of high-quality, low-interest savings for banks, and the payroll service can increase bank deposits and obtain capital reserves.
- The payroll service will generate more traffic for banks, increasing the number of new active customers, creating subsequent opportunities such as newly added credit card customers, revenue from various types of credit services (including money transfers and SMS notifications), sales of investment products and even attracting loan customers to banks.
- The payroll service, and its accompanying value-added services, can generate ongoing fee income for banks, such as account management fees, corporate e-banking service fees and SMS fees. Meanwhile, individuals with payroll cards also generate income for banks, including annual fees, production fees, and escrow fees on their accounts.
- With the payroll service, banks can also develop more business and provide extended services, including providing a full range of services for enterprises and employees, such as collecting information on special deductions, to improve user experience.

Traditionally, the payroll service is just a kind of financial intermediary business, and many individual customers immediately withdraw their money or make an outward transfer upon receipt of the payroll. In this case, banks do not have access to low-cost and high-quality deposits, and cannot benefit from the extended services, so it's important to properly manage the payroll service. In the past, account manager introduction, over-the-counter business referrals, and advertising brochures have been used to improve the fund retention. With the continuous advancement of IT technology, more and more commercial banks are using IT technology and even AI capabilities to build a capital flow prediction system in order to improve the banks' post-management ability of financial intermediary business, accurately control the capital flow, satisfy customers, and improve competitive advantage.

As one of the world's top 20 financial institutions, a joint-stock commercial bank has also been actively developing financial intermediary businesses including payroll service. As the bank's digital services continue to upgrade, a commercial bank with a large number of users has also built an efficient financial big data platform around customer information, account information, capital flow information and other business data, with the help of Intel, Cloudera and other partners. And on this basis, they have developed and deployed AI applications such as capital flow prediction, business recommendation and active marketing through the introduction of clustering algorithms, XGBoost, and other machine algorithms.

As the size of the bank's business, particularly the number of individual customers, has grown, the amount of data involved in its financial intermediary businesses has increased significantly. At the same time, the large number of non-linear, strongly correlated and tightly coupled features of credit cards, personal finance, credit loans and other businesses that are associated with the payroll service also make the complexity of capital flow prediction rise sharply.

In order to effectively respond to these changes, the commercial bank, which has always been highly sensitive to FinTech and digital innovation, together with Intel, explored how to build deep learning methods on the increasingly mature Cloudera financial big data platform and apply them to capital flow prediction scenarios. To better implement this application, Intel has provided Analytics Zoo, an end-to-end unified big data + AI platform, in addition to the powerful computing resources and software tuning methods for the deep learning methods in the solution.

Solution Architecture and Deployment Results

From the bank's point of view, the payroll service is not only an important source of deposits and capital flows, but also an important entry point for the development of other high-quality business, such as credit cards, wealth management and personal loans. Therefore, the bank wants to predict the flow of the user's money within three days after the payroll payment by using deep learning methods, so that it can develop more effective product marketing programs based on different user behavior characteristics.

After thorough communication and exchange of ideas, the requirements for the design, building and deployment of the new prediction solution are summarized as follows:

- The new solution needs to be seamlessly integrated with the bank's existing Cloudera big data platform.
- The new solution can effectively utilize the computing power of the Intel® processors in its IT systems.
- The new solution is flexible enough to use different deep learning frameworks for different scenarios. For example, its original recommender system uses TensorFlow, but in the prediction system, both parties plan to use PyTorch, so the solution needs to provide good support for different frameworks.
- The new solution for the prediction system needs to achieve a prediction accuracy of 80% (AUC value).

Based on the above requirements, as shown in Figure 2-8-7, Intel, together with the bank, planned and designed a deep learning architecture on the Cloudera big data platform, the core of which is to build a high-speed connection between the bank's existing Hadoop/Spark big data platform and the deep learning-based AI applications by introducing the open source Analytics Zoo platform.

With the help of Analytics Zoo, the solution builds five main capabilities on top of the Cloudera big data platform: a deep learning framework, a machine learning framework, a model training/incremental learning cluster, a model distributed inference service, and an AutoML framework.

First, for the distributed architecture of the big data platform, the Analytics Zoo platform enables distributed workflow through the introduction of BigDL and the model distributed inference services. Second, the Pytorch framework used in the prediction system can be deployed directly into the Analytics Zoo platform. In addition, the Analytics Zoo platform also provides the new system with a set of easy-to-use abstractions and APIs, such as transfer learning support, signature operations, Spark data frames and online model service APIs, for conducting model training and inference.

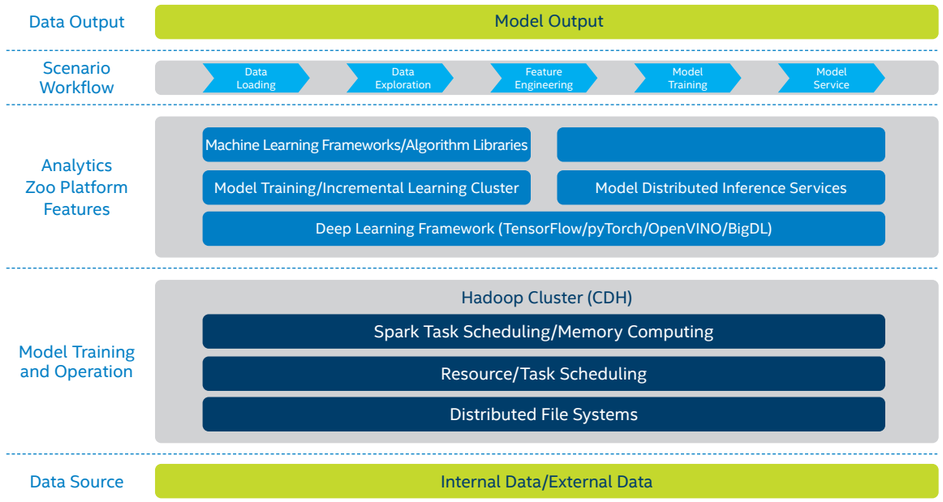


Figure 2-8-7 Deep Learning on a Big Data Platform in a Commercial Bank

To better leverage the computing power of the Intel® processors in the bank's existing IT systems, the Analytics Zoo platform provides acceleration through built-in libraries to improve the inference performance of its prediction system.

To enable users to quickly build the deep learning-based prediction system on the Analytics Zoo platform, Intel facilitated the use of the PyTorch framework in order to refactor the prediction system code on the platform using the MLP model, and performed multiple rounds of iterative optimization based on test results. As shown in Figure 2-8-8, after two rounds of optimization, the predicted effect (AUC value) reaches 87%[38], which satisfies the user's expectation.

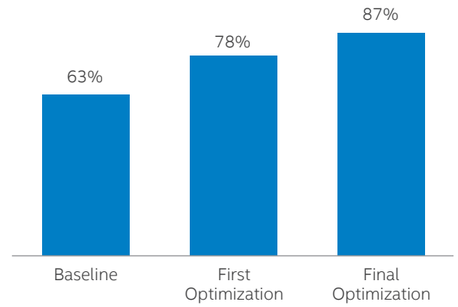


Figure 2-8-8 Optimized AUC Value for a Commercial Bank

Recommended Hardware and Software Configuration

The above capital flow prediction solution can be built with reference to the following Intel® architecture-based platform, with the following environment configuration:

Hardware Configuration

Name	Specification
Processor	2S Intel® Xeon® Gold 6248 Processor or higher
Base Frequency	3.00GHz (Turbo Boost @ 4.0GHz)
Cores/Threads	24/48
HT	On
Turbo	On
Memory	96G (8G DDR4 2666MHz x12) (or configured according to specific data size and model size)
Storage	1TB (or configured according to specific data size and model size)

Software Configuration

Name	Specification
Operating System	CentOS Linux release 7.6.1810/Ubuntu16.04.6
Linux Kernel	3.10.0-957.10.1.el7.x86_64
Workloads	CNN MLP
Compiler	GCC 4.8.5
Framework	PyTorch/Analytics Zoo 0.9.dev
Hadoop Version	CDH5.15/CDH6.2.0

Conclusion

The capital flow management occupies an important position in the operation of commercial banks, and is an important driver for banks to maximize their profitability and one of the tools to prevent and control risks. Intel has learned valuable experience in working with customers to explore the seamless transition from big data to deep learning in the financial industry by collaborating on the development of a deep learning-based capital flow prediction solution. During this process, a series of workflows, methodologies, and application practices around the Analytics Zoo platform have proven to be effective in improving the efficiency of the financial industry in exploring deep learning methods on big data platforms, and can provide good predictive results.

It is a common goal of Intel and its partners to help financial institutions leverage their existing big data platforms and IT resources to deploy deep learning-based AI applications in more financial business scenarios. In the future, Intel plans to work with more partners to further unleash the power of advanced IT products and technologies to better tap into the value of the financial digitalization.

Existing use case solutions all use the Intel® Xeon® processor/Intel® Xeon® scalable processor platform. In order to obtain better image analysis results, it is recommended that users choose the 2nd Generation Intel® Xeon® Scalable Processor with better performance and more optimization options in the AI field to build their solutions.

[38] Test configuration: Processor: 2S Intel® Xeon® Gold 6248R processor, 3.00GHz, 24 cores/48 threads; Memory: 8GB DDR4 2666 x12; Operating System: CentOS Linux release 7.6.1810/Ubun16.04.6; Linux Kernel: 3.10.0-957.10.1.el7.x86_64; Compiler: GCC 4.8.5; Hadoop Version: CDH5.15/CDH6.2.0; Model: CNN MLP; Framework Version: PyTorch/Analytics Zoo 0.9.dev

Technologies

2nd Generation Intel® Xeon® Scalable Processor

The 2nd Generation Intel® Xeon® Scalable Processor is designed specifically for data center modernization, and it provides 25%-35% higher performance[1] than its previous generation. With many new features, improved flexibility and security, and enhanced memory performance, it helps users improve the operational efficiency of various infrastructures, enterprise applications and technical computing, and deliver agile services with enhanced performance and more valuable capabilities that lead to improved total cost of ownership (TCO) and higher productivity.

The Intel® Xeon® Gold 6200 processor series, especially the mainstream Intel® Xeon® Gold 6248 processor, Intel® Xeon® Gold 6240 processor and Intel® Xeon® Gold 6230 processor, are the mainstay of the Intel® Xeon® scalable processor platform. With support for the higher memory speeds, enhanced memory capacity, and four-socket scalability, it delivers significant improvement in performance, advanced reliability, and hardware-enhanced security. It is optimized for demanding mainstream data center, multi-cloud compute, and network and storage workloads, and it is suitable for more complex and diversified application scenarios. In addition, the Intel® Xeon® Gold 6200 series supports dual FMA channels for the first time, which means that the FMA performance has improved by 2 times[2].

Moreover, the 2nd Generation Intel® Xeon® Scalable Processor integrates Deep Learning Boost (Vector Neural Network Instruction, VNNI), expands Intel® AVX-512, and empowers the platform with more and stronger AI capabilities to accelerate AI and deep learning inference and optimize workloads. This gives it a CPU architecture that integrates AI acceleration capabilities. Based on this architecture, most inference tasks are integrated into workloads or applications, providing users with accelerated performance and higher flexibility. In the data centric era, it enables the seamless switch between the multi-cloud and the intelligent edge, and facilitates the AI development and application.

As the "core" of the next generation Xeon® scalable platform, the 2nd Generation Intel® Xeon® Scalable Processor supports the brand new Intel® Optane™ Persistent Memory. When used with the 2nd Generation Intel® Xeon® Gold and Platinum processors, Intel® Optane™ Persistent Memory complements DRAM to significantly improve system performance and accelerate workload processing and service delivery. (For details, please refer to the section "Intel® Optane™ Persistent Memory").

Features:

- Higher Per-Core Performance: Up to 56 cores (9200 series) and up to 28 cores (8200 series), delivering high performance and scalability for compute-intensive workloads across compute, storage, and network usages.
- Intel® DL Boost with VNNI: It brings enhanced artificial intelligence inference performance on CPU. With up to 30X performance improvement over the previous generation[3], it facilitates delivering AI readiness and application from the data center to the edge.
- Industry-leading memory and storage support, greater memory bandwidth/capacity: Support for Intel® Optane™ Persistent Memory, supporting up to 36 TB of system memory capacity when combining with traditional DRAM; 50% increased memory bandwidth and capacity[4]. Support for six memory channels and up to 4 TB of DDR4 memory, per socket, with speeds up to 2933 MT/s (1 DPC). It also supports Intel® Optane™ SSDs and QLC 3D NAND SSDs. For data-intensive workloads, breakthrough memory and storage memory innovations can significantly improve their efficiency and performance.
- Intel® Infrastructure Management Technologies (Intel® IMT): This framework for resource management combines multiple Intel capabilities that effectively support platform-level detection, reporting and configuration.
- Intel® Security Libraries for Data Center (Intel® SecL-DC): This set of software libraries and components enables Intel hardware-based security features.

As an innovation of the Xeon® platform, disruptive by design, the 2nd Generation Intel® Xeon® Scalable Processor sets a new level of platform convergence and capabilities across compute, storage, memory, network, and security.

[1] https://www.intel.com/content/www/us/en/technology-provider/products-and-solutions/xeon-scalable-family/2gen-data-centric-computing-article.html
[2, 3, 4] https://www.intel.com/content/www/us/en/products/docs/processors/xeon/2nd-gen-xeon-scalable-processors-brief.html

2nd Generation Intel® Xeon® Scalable Processor for AI Application

*Supported on select processors only.

	Intel® Xeon® Gold Processor (6200 Series)	Intel® Xeon® Gold Processor (6200 Series)						Intel® Xeon® Platinum Processor (8200 Series)	Intel® Xeon® Platinum Processor (9200 Series)
		Intel® Xeon® Gold 6230 Processor	Intel® Xeon® Gold 6230R Processor	Intel® Xeon® Gold 6240 Processor	Intel® Xeon® Gold 6240R Processor	Intel® Xeon® Gold 6248 Processor	Intel® Xeon® Gold 6248R Processor		
Pervasive Performance and Security									
Highest Core Count Supported	24 cores	20 cores	26 cores	18 cores	24 cores	20 cores	24 cores	28 cores	56 cores
Highest Supported Frequency	4.4 GHz	3.90 GHz	4.00 GHz	3.90 GHz	4.00 GHz	3.90 GHz	4.00 GHz	4.0 GHz	3.8 GHz
Number of CPU Sockets Supported	Up to 4 sockets	Up to 4 sockets	Up to 2 sockets	Up to 4 sockets	Up to 2 sockets	Up to 4 sockets	Up to 2 sockets	Up to 8 sockets	Up to 2 sockets
Intel® Ultra Path Interconnect (UPI)	3	3	2	3	2	3	2	3	4
Intel® UPI Speed	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s	10.4 GT/s
Intel® Advanced Vector Extensions 512 (Intel® AVX-512)	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA	2 FMA
High-test Memory Speed Support (DDR4)	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s	2933 MT/s
Highest Memory Capacity Supported Per Socket*	1 TB, 2 TB, 4.5 TB	1 TB	1 TB	1 TB	1 TB	1 TB	1 TB	1 TB, 2 TB, 4.5 TB	3.0 TB
16 Gb DDR4 DIMM Support	●	●	●	●	●	●	●	●	●
Intel® Deep Learning Boost (Intel® DL Boost) with Vector Neural Network Instruction (VNNI)	●	●	●	●	●	●	●	●	●
Intel® Optane™ Persistent Memory Module Support*	●	●	●	●	●	●	●	●	
Intel® Omni-Path Architecture (Discrete PCIe* card)	●	●	●	●	●	●	●	●	●
Intel® QuickAssist Technology (Integrated in chipset)	●	●	●	●	●	●	●	●	
Intel® QuickAssist Technology (Discrete PCIe* Card)	●	●	●	●	●	●	●	●	●
Intel® Optane™ SSDs	●	●	●	●	●	●	●	●	●
Intel® SSD Data Center Family (3D NAND)	●	●	●	●	●	●	●	●	●
PCIe 3.0	●	●	●	●	●	●	●	●	●
Intel® QuickData Technology (CBDMA)	●	●	●	●	●	●	●	●	●
Non-Transparent Bridge (NTB)	●	●	●	●	●	●	●	●	●
Intel® Turbo Boost Technology 2.0	●	●	●	●	●	●	●	●	●
Intel® Hyper-Threading Technology (Intel® HT Technology)	●	●	●	●	●	●	●	●	●
Node Controller Support	●	●	●	●	●	●	●	●	●

*Supported on select processors only.

For more information on the 2nd Generation Intel® Xeon® Scalable Processor, please visit:
https://www.intel.com/content/www/us/en/products/processors/xeon/scalable.html

Intel® Optane™ Persistent Memory

Intel's breakthrough product, Intel® Optane™ Persistent Memory, is disrupting the traditional memory-storage hierarchy by creating a new tier to fill the memory-storage gap, and providing a large persistent memory hierarchy at a reasonable price, which can provide greater performance, efficiency, and affordability for memory-intensive workloads, virtual machine density and fast storage. Then it facilitates accelerating IT transformation through faster analysis, cloud services, artificial intelligence training and inference, and next generation communication services than ever before to meet the needs of the data era.

Intel® Optane™ Persistent Memory combines cost, capacity, non-volatility and performance features, and a single module provides 128/256/512 GiB options available and is compatible with DDR4 socket. When applied on the platform based on the 2nd Generation Intel® Xeon® Scalable Processor together with the traditional DDR4 DRAM memory, it can build a capacity of up to 24TiB on the eight socket system at lower cost (each socket providing up to 3TiB of Optane persistent memory), so as to allow users to load datasets on the memory system close to the processor that are far larger than previous datasets, meet the application load needs with demanding requirements on the large memory including memory database, larger-scale dataset analysis, and AI inference and iteration, and enable more data processing and analysis to be carried out in real time.

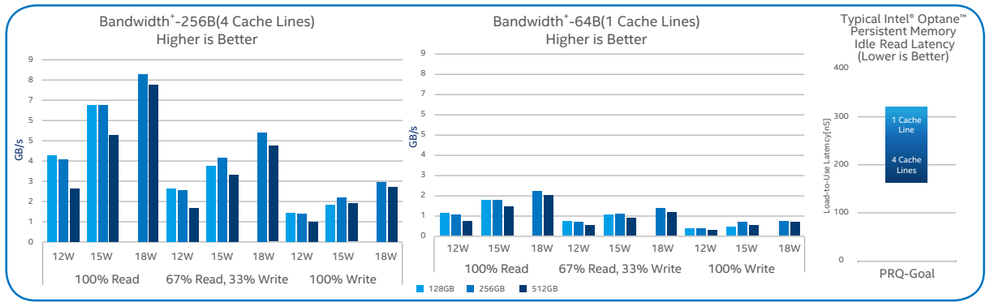
Based on the combination of memory and storage features, Optane™ persistent memory has two operating modes: Memory Mode and App Direct Mode. With distinct operating modes, customers have the flexibility to significantly improve system performance across multiple workloads.

Memory Mode. Memory Mode is great for large memory capacity and does not require application changes. In Memory Mode, the CPU memory controller uses Intel® Optane™ Persistent Memory as addressable main memory to extend the amount of available volatile memory visible to the Operating System, while using the DRAM as cache. This can directly facilitate virtualization and container technology because it can increase the density of virtual machines or containers on a single physical server at lower cost, or provide larger memory capacity for each virtual machine and container without rewriting software.

App Direct Mode. In App Direct Mode, the operating system treats DRAM and Intel® Optane™ Persistent Memory as two separate pools of memory. It is byte addressable like memory, persistent like storage. The power of persistent memory adds business resilience to systems with faster restart times and lower cost because data is retained even during maintenance or power cycles.

Mixed Mode. It is a subset of App Direct and can be provisioned so that some of Intel® Optane™ Persistent Memory is in Memory Mode and the remaining is in App Direct Mode, thus meeting the needs of workloads or application environments with dual demands.

Intel® Optane™ Persistent Memory is launched with other Intel® Xeon® scalable processor platform products and optimized for the 2nd Generation Intel® Xeon® Scalable Processor. Its unique application advantages have attracted many ISV partners to start tuning their related software from the very beginning of its launch. Meanwhile, there are also cloud infrastructure and data analysis users who have also adopted Intel® Optane™ Persistent Memory. These partners and users have initially witnessed the application value of Intel® Optane™ Persistent Memory in multiple modes in the corresponding application tuning and application practice.



Product Family	Intel® Optane™ Persistent Memory					
Form Factor	Persistent Memory Module (PMM)					
PMem SKU[1]	128 GiB		256 GiB		512 GiB	
User Capacity	126.4 GiB[8]		252.4 GiB[8]		502.5 GiB[8]	
MOQ	4	50	4	50	4	50
MM#	999AVV	999AVW	999AVX	999AVZ	999AW1	999AW2
Product Code	NMA1XXD128GPSU4	NMA1XXD128GPSUF	NMA1XXD256GPSU4	NMA1XXD256GPSUF	NMA1XXD512GPSU4	NMA1XXD512GPSUF
Model String	NMA1XXD128GPS		NMA1XXD256GPS		NMA1XXD512GPS	
TECHNOLOGY	Intel® Optane™ Technology					
Limited Warranty	5 years					
Annual Failure Rate (AFR)	≤ 0.44					
Endurance 100% Write 15W 256B	292 PBW		363 PBW		300 PBW	
Endurance 100% Write 15W 256B	91 PBW		91 PBW		75 PBW	
Endurance 100% Read 15W 64B	6.8 GB/s		6.8 GB/s		5.3 GB/s	
Endurance 100% Write 15W 256B	1.85 GB/s		2.3 GB/s		1.89 GB/s	
Endurance 100% Read 15W 64B	1.7 GB/s		1.75 GB/s		1.4 GB/s	
Endurance 100% Write 15W 64B	0.45 GB/s		0.58 GB/s		0.47 GB/s	
DDR Frequency	2666, 2400, 2133, 1866 MT/s					
Maximum Thermal Design Power (TDP)	15W		18W			
Temperature (TJMAX)	≤ 84℃ (85℃ shutdown, 83℃ default) media temperature					
Temperature (Ambient)	10W: 54℃ @ 2.4m/s					
Temperature (Ambient)	12W: 49℃ @ 2.4m/s					
Temperature (Ambient)	15W: 44℃ @ 2.7m/s					
Temperature (Ambient)	N/A	18W: 40℃ @ 3.7m/s				

Note: [1]GiB = 2^{30}; GB = 10^9

For more information, please refer to
https://www.intel.com/content/www/us/en/products/memory-storage/optane-dc-persistent-memory.html

Intel® Optane™ SSDs and Intel® SSDs with Intel® QLC 3D NAND Technology

Intel® Optane™ SSDs and Intel® SSDs with Intel® QLC 3D NAND Technology, with their innovative storage hierarchy, facilitates building a future-ready data center to accelerate changes and achieve great leaps.

As a high-end member of Intel's SSD product line, Intel® Optane™ SSDs feature innovative 3D XPoint™ storage media and incorporate advanced system memory controllers, interface hardware, and software technologies, delivering low latency and high stability. It is designed to alleviate data center storage bottlenecks and allow for bigger, more affordable datasets, thus accelerating applications, reducing transaction costs for latency-sensitive workloads, and improving data center TCO. Intel® Optane™ SSDs, with more comprehensive, better and more balanced IT infrastructure capability, can undoubtedly bring higher efficiency to data-intensive AI model training and inference. Take Intel® Optane™ SSD DC P4800X as an example. It offers up to 550,000 IOPS of random read/write capacity and less than 10 ms of read/write latency, enabling the solution to perform more effectively in multi-user and high-concurrency scenarios. Meanwhile, its Drive Writes Per Day (DWPD) is as high as 60, which is much higher than NAND SSDs, so it can give the storage system a longer lifespan and ensure better value.

Intel® SSDs use breakthrough, trusted 3D NAND technology to improve storage, thus providing a cost-effective replacement for traditional hard disk drives (HDD), and helping customers improve user experiences, increase the performance of applications and services and reduce TCO. As a "new force" in SSDs, the Intel® SSD D5-P4320 series rely on Intel's leading 64-layer 3D NAND technology to enable a single QLC SSD disk capacity of up to 7.68 TB (Terabytes) in order to adequately fulfill the storage requirements of infrastructure users such as data centers for massive data. It also has a random read IOPS of up to 427,000, and when paired with the 2nd Generation Intel® Xeon® Scalable Processor, it is especially suitable in terms of meeting "Write Once, Read Many" (WORM) performance requirements in application scenarios, such as AI training, providing a storage framework with high efficiency, high stability and low energy consumption to support complex and diverse workloads.

To find out more, visit:

- https://www.intel.com/content/www/us/en/products/memory-storage/optane-memory/optane-memory-h10-solid-state-storage.html
- https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds.html

Unified Open Source Big Data Analytics + AI Platform Analytics Zoo

Analytics Zoo is a unified big data analytics + AI open source platform. It is designed for users to develop big data-based and end-to-end deep learning applications.

Analytics Zoo allows users to implement TensorFlow, Keras, Pytorch and BigDL, as well as other frameworks that may need to be supported in the future on top of Apache Spark/ Flink and Ray, seamlessly integrate them into an pipeline, and transparently scale out these models to large data clusters with hundreds or thousands of nodes for distributed training or inference, thereby further simplifying the development of AI solutions without additional dedicated infrastructure.

In order to improve computing performance, Analytics Zoo integrates various software libraries, such as Intel® MKL and Intel® MKL-DNN. In terms of hardware, it is based on the Intel® Xeon® processor platform and fully releases the integrated vector and deep learning instructions of the 2nd Generation Intel® Xeon® Scalable Processor, which can greatly improve the training and inference speed.

The benefits of integrating data storage and processing pipelines into a unified infrastructure without moving data is obvious - it can not only improve development and deployment efficiency and scalability, reduce hardware management and developers' time to learn new languages, improve development and deployment efficiency, resource utilization and flexibility, but also reduce total cost of ownership without affecting computing efficiency and performance. Developers just need to make full use of the rich features and functions provided by Analytics Zoo and various analytics and AI tools when scaling up their AI solutions, so as to achieve the efficient integration, deployment and application of big data analytics and AI.

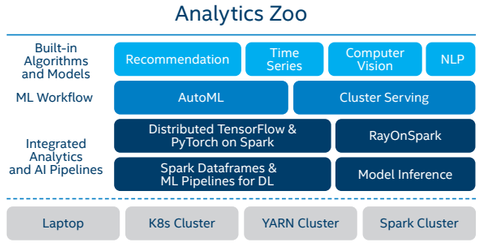
Among the many AI frameworks supported by Analytics Zoo platform, BigDL is an open source platform developed by Intel itself. BigDL is a distributed deep learning framework based on Apache Spark, which can run seamlessly and directly on top of existing Apache Spark and Hadoop clusters without any modification to the clusters. Based on BigDL, developers can write deep learning applications as Scala or Python programs and take advantage of the power of scalable Spark clusters to promote the integration of big data analytics and AI. Users who have been familiar with and enabled BigDL in the past few years can invoke BigDL directly through Analytics Zoo for seamless switching.

Technical Features of Analytics Zoo:

- End-to-end pipeline, applying AI models (TensorFlow, PyTorch, OpenVINO™ toolkit, and more) to distributed big data:
  - Distributed training and prediction with Spark code for TensorFlow or PyTorch
  - Native deep learning support (TensorFlow/Keras/ PyTorch/ BigDL) in the Spark ML pipeline
  - With RayOnSpark, run Ray programs directly on big data clusters
  - Provide Plain Java/Python APIs (TensorFlow/PyTorch/ BigDL/OpenVINO™) to serve Model Inference
- Highly abstract ML workflows to automate machine learning tasks
  - Cluster Serving for automated distributed (TensorFlow/ PyTorch/Caffe/OpenVINO™) model inference
  - Extensible AutoML for time series-based data analysis and prediction
- Built-in models for recommender systems, time series analysis, computer vision and natural language processing applications

Why Analytics Zoo:

- AI models (e.g., TensorFlow, Keras, PyTorch, BigDL, OpenVINO™ toolkit) can be easily applied to distributed big data.
- AI applications can be scaled transparently from a single laptop to large clusters with "zero" code changes.
- AI pipelines can be deployed to existing YARN or K8S clusters without making any changes to the cluster.
- The process of applying machine learning (e.g. feature engineering, hyper-parameter tuning, model selection, distributed inference) can be automated.



To find out more, visit:
https://software.intel.com/content/www/us/en/develop/blogs/analytics-zoo-unifying-analytics-ai-for-apache-spark.html?elq_cid=4287274&erpm_id=7282583

Intel® Data Analytics Acceleration Library

As a branch of artificial intelligence, machine learning is now gaining tremendous attention and advanced analytics based on machine learning is becoming increasingly popular due to its ability to help IT staff, data scientists, various business teams and their organizations quickly unlock advantages over other analytics. And machine learning offers many new commercial and open source solutions, providing a rich ecosystem for developers. In addition, developers can choose from a variety of open source machine learning libraries such as Scikit-learn, Cloudera and Spark MLlib.

Intel® Data Analytics Acceleration Library (Intel® DAAL)

To help its industry users deploy machine learning, Intel also offers a high-performance and systematic solution that covers a rich set of resources including processors, optimized software and developer support, and a robust ecosystem.

Machine learning requires robust computing power. Intel® Xeon® processors provide a scalable benchmark, which is specifically designed to meet the highly parallel workloads unique to machine learning and its memory and architecture (networking) requirements. In an Intel test, the processor reduced system training time by a factor of 50[1].

In addition, Intel provides software support, including:

- Libraries, languages, and building blocks optimized for Intel® Xeon® processors, oneDNN and **Intel® Data Analytics Acceleration Library (Intel® DAAL)**, and Intel® Distribution for Python.
- Optimization frameworks that simplify development, including Apache Spark, Caffe, Torch, and TensorFlow. Intel supports both open source and commercial software, enabling users to quickly take advantage of the latest processor and system features available on the market.

Intel® DAAL is a suite of end-to-end software solutions designed to help data scientists and analysts quickly build everything from data pre-processing, to data feature engineering, data modeling and deployment. It provides various data analytics needed to develop machine learning and analytics as well as high-performance building blocks required by algorithms. Classical machine learning algorithms such as linear regression, logistics regression, LASSO, AdaBoost, Bayesian classifiers, support vector machines, K nearest neighbors, Kmeans clustering, DBSCAN clustering, various decision trees and random forests Gradient Boosting are now supported. These algorithms are highly optimized to achieve high performance on Intel® processors. For example, a leading big data analytics technology and services provider in China has used these resources to improve the performance of data mining algorithms by 3X to 14X[2].

To make it easier for developers to use Intel® DAAL in machine learning applications in Intel-based environments, Intel has open-sourced the entire project (**https://github.com/intel/daal**), and provides full-memory, streaming and distributed algorithm support for different big data scenarios. For example, DAAL Kmeans can be well combined with Spark to perform multi-node clustering on a Spark cluster. In addition, Intel® DAAL provides interfaces to C++, Java, and Python.

DAAL4py

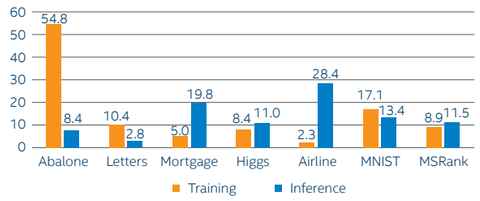
To deliver better support for the most widely used Python application Scikitlearn, Intel® DAAL provides a very simple Python interface DAAL4py (open source resources at **https://github.com/IntelPython/daal4py**), which works seamlessly with Scikitlearn and enables underlying algorithmic acceleration for machine learning. Developers do not need to modify the Scikitlearn code to take advantage of automatic vectorization and multi-threading. DAAL4py currently supports the following algorithms in Scikitlearn:

- sklearn.linear regression, sklearn.ridge regression, logistics regression
- PCA
- KMeans
- pairwise_distance
- SVC (SVM classification)

Intel-optimized XGBoost

XGBoost is a machine learning open source library based on progressive Gradient Boosting, which is widely used in a variety of classification and decision making businesses. To further enhance its performance, Intel has optimized and open-sourced the codebase[3] and the latest optimizations have been integrated into XGBoost 1.0 and later versions. Compared to XGBoost version 0.9, the new version offers a 2X-54X performance increase. [4]

Gradient Boosting Performance (Higher is better) Intel® DAAL 2020 vs DMLC XGBoost 0.9 Speed-Up



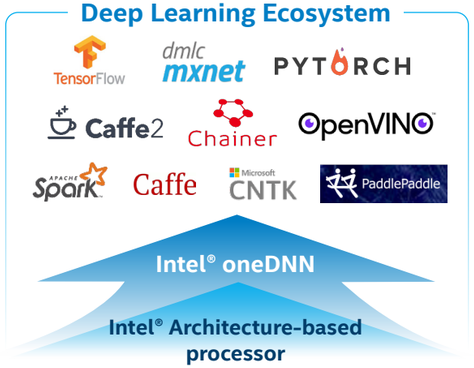
For more information, visit: https://software.intel.com/en-us/daal

[1, 2] https://software.intel.com/content/www/us/en/develop/download/meritdata-case-study.html
[3] Performance optimizations for Intel CPUs : https://github.com/dmlc/xgboost/pull/3957/files
[4] https://software.intel.com/daal

Intel® Deep Neural Network Library (oneDNN)

The Intel® Deep Neural Network Library (formerly the Intel® Math Kernel Library for Deep Neural Networks, Intel® MKL-DNN) is an open source, performance enhancement library for deep-learning applications, and a foundation library created by Intel to help developers make full use of Intel® architecture to promote the research and application of deep learning. (Source Code Address: **https://github.com/intel/mkl-dnn**)

oneDNN, as a performance enhancement library specifically designed for accelerating deep learning framework on Intel® architecture, includes highly vectorized and threaded building blocks for implementing deep neural networks with C and C++ interfaces, and has a broad ecosystem of deep learning research, development and applications. It currently supports: a rich set of deep learning software products such as TensorFlow, PyTorch, MXNet, Caffe, Spark BigDL and OpenVINO™ toolkit.



In order to effectively accelerate deep learning model on Intel® architecture, and improve the efficiency of other performance-sensitive applications in various neural networks, oneDNN provides a number of optimized deep learning operation primitives that can be used in different deep learning frameworks to ensure the efficient implementation of common building blocks. These modules include:

- Matrix multiplication and convolution: 1D/2D/3D, Winograd 2D
- RNN primitives;
- Inner product;
- Pooling: maximum, minimum, average;
- Normalization: local response normalization across channels (LRN), batch normalization;
- Activation: rectified linear unit (ReLU);
- Data manipulation: multi-dimensional transposition (conversion), split, concat, sum and scale.

These efficient function blocks can be applied to a wide range of deep learning models, such as:

Application type	Topologies
Image Recognition	AlexNet, VGG, GoogleNet, ResNet, MobileNet
Image Segmentation	FCN, SegNet, MaskRCNN, U-Net
Volumetric Segmentation	3D-Unet
Object Detection	SSD, Faster R-CNN, Yolo
Machine Translation	GNMT
Speech Recognition	DeepSpeech, WaveNet
Adversarial Networks	DCGAN, 3DGAN
Reinforcement Learning	A3C

In order to greatly improve the performance of deep learning on CPU, Intel has also cooperated with many open source communities to integrate this library into various deep learning frameworks. For example, as early as 2016, Caffe optimized by oneDNN achieved a performance improvement of up to 10X over the original Caffe using Intel® Xeon® Processor E5-2697 v3[1]. In 2019, the optimized ResNet-50 also achieves the leading performance of 7,736 images per second on Intel® Xeon® Platinum 9282 Processor[2].

oneDNN has now become the basic configuration of many deep learning frameworks running on CPU. Developers can directly benefit from the performance improvement brought by oneDNN in the installation and application of the deep learning framework.

To find out more, visit:

- https://software.intel.com/content/www/us/en/develop/articles/intel-mkl-dnn-part-1-library-overview-and-installation.html
- https://software.intel.com/content/www/us/en/develop/articles/intel-mkl-dnn-part-1-library-overview-and-installation.html

[1] https://software.intel.com/es-es/node/604830?language=en
[2] https://www.intel.com/content/dam/www/public/us/en/images/diagrams/rwd/xeon-scalable-max-inference-rwd.png

Intel® Optimization for Caffe

Intel® Optimization for Caffe was integrated with the then current release of Intel® MKL from the original version, and it is specially optimized for Intel® AVX 2 and Intel® AVX-512 which were integrated with Intel® Xeon® processors at that time. It is fully compatible with BVLC Cafe, and incorporates all the advantages of BVLC Cafe. With processor optimization feature, it shows excellent performance on Intel® processors, and supports multi-node distributed program training.

Intel® Optimization for Caffe supports a complete set of Post-training quantization options and has been implemented in a large number of CNN models. Especially in the platform based on the 2nd Generation Intel® Xeon® Scalable Processor (see the processor section), the integrated Intel® Deep Learning Boost (VNNI instruction set) with optimization support for INT8 allows for accelerating the inference speed of multiple deep learning models when using INT8 up to 2-4 times that when using FP32 (see the following figure)[1] without affecting the prediction accuracy, thus greatly improving the working efficiency of users' deep learning applications.

To find out more, visit:

- https://software.intel.com/en-us/articles/caffe-optimized-for-intel-architecture-applying-modern-code-techniques
- https://software.intel.com/content/www/us/en/develop/videos/what-is-intel-optimization-for-caffe.html

Optimized Deep Learning Frameworks and Toolkits

Benefits of ResNet-50 with Intel® Deep Learning Boost

2nd Generation Intel® Xeon® Platinum 8280 Processor vs Intel® Xeon® Platinum 8180 Processor

Intel® Xeon® Scalable Processor	2nd Generation Intel® Xeon® Scalable Processor	mxnet	PYTORCH	TensorFlow	Caffe	OpenVINO
FP32	INT8 W/Intel® DL Boost	3.0x	3.7x	3.9x	4.0x	3.9x
INT8	INT8 W/Intel® DL Boost	1.8x	2.1x	1.8x	2.3x	1.9x

[1]For configuration details, see: https://www.intel.com/content/www/us/en/benchmarks/server/xeon-scalable/xeon-scalable-artificial-intelligence.html

Intel® Optimization for TensorFlow

Intel® Optimization for TensorFlow is an optimized version launched by Intel to deal with the performance challenges of running deep learning models on CPU, making sure that deep learning workloads can run efficiently with Intel® MKL-DNN basic computing units under all conditions.

In order to significantly improve performance, Intel continues to optimize TensorFlow in other ways.

Graph Optimizations

Intel has introduced a number of graph optimizations to replace default TensorFlow operations with Intel optimized versions when running on CPU. This ensures that users can run their existing Python programs and obtain the performance gains without changes to their neural network model. Meanwhile, it can eliminate unnecessary and costly data layout conversions, fuse multiple operations together to enable efficient cache reuse on CPU, and handle intermediate states that allow for faster backpropagation.

These graph optimizations enable greater performance without introducing any additional burden on TensorFlow programmers. In addition, data layout optimization is a key performance optimization. Previously, Intel inserts a data layout conversion operation from TensorFlow's native format to an internal format, performs the operation on CPU and converts operation output back to the TensorFlow format, but these conversions introduce a performance overhead. Now, the sub-graphs that can be entirely executed using Intel® MKL optimized operations can eliminate the conversions within the operations in the sub-graph. Automatically inserted conversion nodes take care of data layout conversions at the boundaries of the sub-graph. Another key optimization is the fusion pass that automatically fuses operations that can run efficiently as a single Intel® MKL operation.

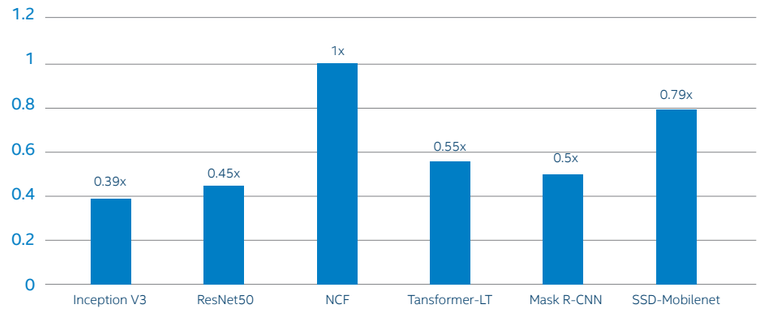
Other Optimizations

Intel has also tweaked a number of TensorFlow framework components to enable the highest CPU performance for various deep learning models. For example, Intel developed a custom pool allocator using existing pool allocator in TensorFlow to ensure that both TensorFlow and Intel® MKL share the same memory pools (using the Intel® MKL imalloc functionality) and Intel doesn't return memory prematurely to the operating system, thus avoiding costly page misses and page clears. In addition, Intel carefully tuned multiple threading libraries (pthreads used by TensorFlow and OpenMP used by Intel® MKL) to coexist and not to compete against each other for CPU resources, thus improving the comprehensive utilization rate of resources.

To find out more, visit:

- https://www.intel.ai/tensorflow/?_ga=2.231295069.330745958.1563951842597697079.1551333838&elq_cid=4287274&erpm_id=7282583
- https://www.intel.ai/improving-tensorflow-inference-performance-on-intel-xeon-processors/#gs.v0kayg

Results of Latency (Intel® oneDNN/Eigen Library) - Lower is Better



Latency Performance of TensorFlow Based on Intel® oneDNN[1]

[1] System Configuration: Intel® Xeon® Platinum 8180 Processor @2.50 GHz; OS: CentOS Linux 7 (Core); TensorFlow Source Code: https://github.com/tensorflow/tensorflow; TensorFlow Commit ID: 355cc566efd2d86fe71fa9d755ceabe546d577a

Intel® Distribution for Python

Intel® Distribution for Python is a powerful software development tool kit developed by Intel. It provides everything needed to write Python native extensions, including C and Fortran compilers, math libraries and analyzers, and integrates multiple high-performance data analysis and math libraries, such as NumPy, SciPy, Scikit-learn, Pandas, Jupyter, matplotlib and mpi4py.

Intel® Distribution for Python is one of the important toolsets of Intel® Parallel Studio XE, with many features and high efficiencies:

- Accelerate compute-intensive applications including numbers, science, data analysis, and machine learning by providing out-of-the-box tools such as uMath, NumPy, SciPy and Scikit-learn.
- Integrated with Intel® performance library (such as Intel® MKL); built-in latest vectorization instructions, such as Intel® AVX-512 and multithreading instructions, Numba and Cython; access composable parallelism with multithreading building blocks library Intel® TBB; unlock Python's parallel application function based on multi-core processor, thus improving the performance of Python program running on platforms based on Intel® architecture, and ensuring good system compatibility without any code changes.
- Support Python2.7, Python3.6 and the latest generation of Intel® processors. It provides optimized deep learning libraries and machine learning libraries such as TensorFlow, Caffe, support vector machine (SVM), K-means prediction, random forests and XGBoost algorithms, so as to construct and expand production ready algorithms for workloads such as scientific computing and machine learning.

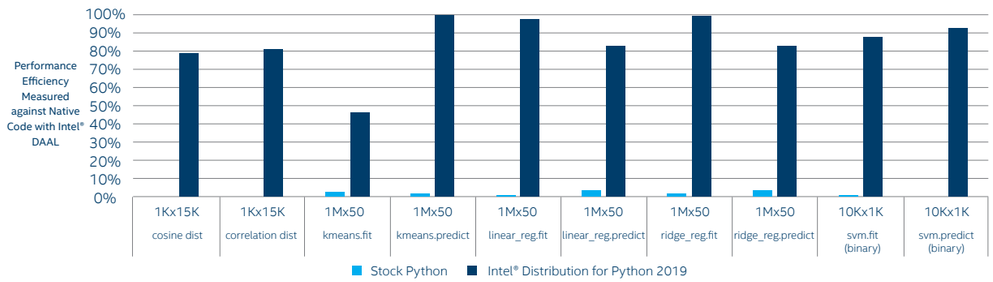
The benchmark tests compare the efficiency of the Intel® Distribution for Python with the Scikit-learn toolkit (a toolkit widely used for numerical calculation, scientific calculation and machine learning) in other open source Python, and show that the performance indicators of Intel® Distribution for Python have significantly improved (see the figure below. The higher the efficiency, the faster the function and the closer to native C speed). For example, the efficiency of algorithms such as K-means clustering and linear regression in Intel® Distribution for Python can reach 90% of the efficiency of C in Intel® DAAL.

Simple deployment and ease of use are also one of the major features of Intel® Distribution for Python - by using Conda Package Manager and Anaconda Cloud, users can install the core Python environment with only one command:

- conda install intelpython3 -c intel

In addition, the Intel® Distribution for Python is a pre-built binary file and can also be obtained through various channels such as pip, Docker images, YUM and APT repos.

To find out more, visit:
https://software.intel.com/en-us/distribution-for-python



Intel's Optimizations Improve Python Scikit-learn Efficiency Closer to Native Code Speeds on Intel® Xeon® Processors

Intel® Distribution for PyTorch

PyTorch is an open source deep learning library designed to help data scientists and developers improve the efficiency of deep learning training and inference, with a high degree of flexibility, ease of use and high speed of training and inference, and is also very popular in the industry.

PyTorch offers a number of excellent features based on its simple, flexible architecture. For example, it provides automatic derivation, which makes model building simple and fast; it uses dynamic models, which can be tuned at any part of the execution process, greatly improving ease of use; it can automatically calculate gradients and update network parameters; it provides flexible training methods, allowing users to customize the communication methods to implement new communication algorithms; it provides a hybrid Python and C++ front-end, which can be used for training in Python front-end and deployment in C++ front-end. In addition, PyTorch is supported by a strong community and rich resources.

To achieve high performance and efficiency on CPUs, Intel introduced the Intel® Distribution for PyTorch. This optimized version leverages Intel® MKL, which enables it to take full advantage of the CPU's parallel computing power and vectorization techniques to optimize matrix multiplication performance, and Intel® MKL-DNN, which enables it to maximize the performance of computing commonly used in deep learning, such as convolution and normalization by utilizing CPU's parallel computing power and vectorization techniques and the efficient use of the CPU's on-chip cache. For the computational graph characteristics at hierarchical or node-level of deep learning networks, Intel optimizes various layers at the node level, such as convolution, matrix multiplication, ReLU and pool, to minimize data transfer and ensure efficient use of SIMD instructions, execution units, registers, and memory cache hierarchies; at the computational graph level, Intel uses various data ordering policies and layer fusion to optimize groups of nodes, such as fusing ReLU to convolution so that data still performs ReLU operations at the last convolution cycle in the registers. In addition, Intel® MKL-DNN is integrated into the PyTorch back-end by leveraging the key DNN layer of Intel® MKL-DNN API.

With multiple in-depth optimizations, when running on the Intel® Xeon® Platinum 8280 processor, Intel® Distribution for PyTorch has been tested to deliver 7.7X, 47X and 23.6X performance improvements on ResNet50, Faster R-CNN and RetinaNet, respectively.[1]

In addition, when using Intel® Distribution for PyTorch, users are often not required to modify the original Pythorch scripts and codes.

To find out more, visit:
https://software.intel.com/content/www/us/en/develop/articles/intel-and-facebook-collaborate-to-boost-pytorch-cpu-performance.html?wapkw=Pytorch&elq_cid=4287274&erpm_id=7282583

[1] https://software.intel.com/content/www/cn/zh/develop/articles/intel-and-facebook-collaborate-to-boost-pytorch-cpu-performance.html?wapkw=Pytorch&elq_cid=4287274&erpm_id=7282583

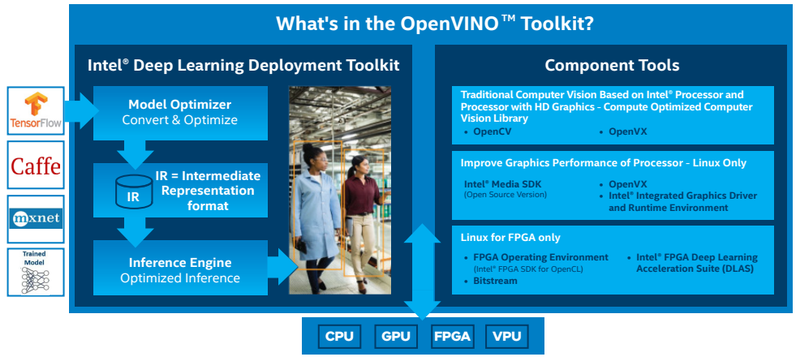
OpenVINO™ Toolkit

OpenVINO™ Toolkit is a software toolkit introduced by Intel to accelerate deep learning inference and deployment, which is used to accelerate high-performance computer vision processing and application. The tool allows for heterogeneous execution, supports Windows and Linux systems, and Python/C++. It can effectively promote the in-depth application of computer vision technology in fields from intelligent cameras, video surveillance, robots, to intelligent transportation and intelligent healthcare.

This toolkit is designed to increase performance and reduce development time for computer vision solutions. It simplifies access to the benefits from the rich set of hardware options available from Intel which can increase performance, reduce power, and maximize hardware utilization – letting you do more with less and opening new design possibilities.

By extending workloads across Intel® hardware (including accelerators) based on deep convolutional neural network (CNN), the OpenVINO™ toolkit can rely on chips such as Integrated GPU with Intel® processors, FPGA and Intel® Movidius™ VPU to enhance the features and performance of the vision system. The newly released version of OpenVINO™ has been able to support the 2nd Generation Intel® Xeon® Scalable Processor, and improve inference performance by using Intel® AVX-512 and Intel® DL Boost with VNNI. It can help customers quickly complete hardware product upgrade and algorithm migration without changing software, thus helping them accelerate the development of high-performance computer vision and deep learning applications on the edge:

- Increase deep learning performance related to computer vision up to 19X on Intel® architecture platform[1];
- Release the performance bottleneck of CNN-based network in edge devices;
- Accelerate and optimize the traditional API implementation of OpenCV and OpenXV visual libraries;
- Run on devices including CPU, GPU, FPGA based on common API interface;
- The OpenVINO™ toolkit optimized for Intel® platform mainly includes two parts: deep learning deployment toolkit and traditional computer vision toolkit. The deep learning deployment toolkit includes two core components: Model Optimizer and Inference Engine;
- Model Optimizer: converts a given model into a standard Intermediate Representation (IR), and optimizes the model. Support for deep learning frameworks including ONNX, TensorFlow, Caffe, MXNet, Kaldi;
- Inference Engine: supports accelerated operation of deep learning model at hardware instruction set level, and supported hardware devices include: CPU, GPU, FPGA, VPU.



Provide a cross-platform tool to support computer vision and deep learning inference acceleration

Meanwhile, the traditional OpenCV image processing library has also been optimized with instruction set, achieving significant improvement in performance and speed. Computer vision tools include:

- OpenCV: Precompiled OpenCV and the new Intel® Photography Vision Library, with features such as face detection/recognition, blink detection and smile detection;
- OpenVX: Graphics-based OpenVX implementation that supports traditional CV operations and CNN primitives. Supports Khronos OpenVX neural network extension 1.2;
- Miscellaneous: Include OpenCL™ driver and runtime libraries, and media drivers to simplify computer vision SDK working with Intel® Media SDK and Intel® SDK OpenCL™ applications. Intel® MKL- DNN and CLDNN are included and do not need to be downloaded separately.

In addition, as a toolkit designed to quickly and effectively implement computer vision and deep learning in multiple applications, the OpenVINO™ toolkit optimized for Intel® platform currently provides MO files of pre-converted Caffe, TensorFlow and MXNet models, including VGG-16, VGG-19, Squeezenet, ResNet, Inception, CaffeNet, SSD, Faster-RCNN and FCN8, and has more than 100 pre-trained models. Software developers and data scientists can use these tools to quickly build personalized deep learning applications, and can use the basic libraries of OpenCV and OpenVX to create specific algorithms and develop customized and innovative applications.

The OpenVINO™ toolkit has made vision a reality on Intel platforms and has helped many users easily develop and rapidly deploy computer vision applications, demonstrating the great potential of AI solutions in a variety of deep learning application scenarios.

To find out more, visit:
https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

[1] https://software.intel.com/en-us/articles/a-guide-for-setting-up-docker-based-openvino-development-environment-with-ubuntu-system

Intel® Software Guard Extensions (Intel® SGX)

Intel® Software Guard Extensions (Intel® SGX) is a new set of instruction set extension and access control mechanism that support Intel® Xeon® processors E3-1500 v5 and v6, Intel® Xeon® processor E2100 series, Intel® Core™ processor family from 6th generation onwards, and Intel® Celeron® processors J4105 or J4005 (BIOS model with Intel® SGX support). It is designed to effectively prevent application code and data breach or tampering through hardware-based isolation and memory encryption. With Intel® SGX, developers can place applications and sensitive data in "enclaves" within specific hardware to achieve hardware-assisted confidentiality and integrity protection and effectively block access from processes with higher privilege, thereby enhancing security levels and providing several features:

- Enhanced confidentiality and integrity protection
- Remote authentication & supply
- Low learning curve
- Significantly reducing the attack surface

Pushing application security limits

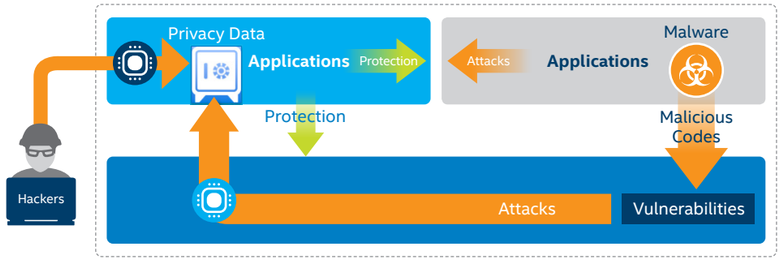
For a long time, developers have been limited by the security features developed and configured by the platform provider for applications. The Intel® SGX now uses a new model that leverages platform and Operating System (OS) strengths to improve security, and allows developers to understand and decide which applications and confidential data need additional protection, thereby unlocking the new performance benefits of chip-level security technologies and effectively helping applications protect themselves based on developer settings.

Providing a new approach to security

To address many security vulnerabilities and system threats, Intel has developed a hardware-based Trusted Execution Environment (TEE) to reduce the attack surface. Intel® SGX provides new Intel® architecture instructions that allow applications to use them to partition areas for greater privacy, as well as to select code and data to prevent direct attacks on code or data stored in memory while it is executing.

Enhancing the effectiveness of federated learning

During federated learning performed on Intel® SGX, AI models and training data are deployed in protected hardware enclaves, which dramatically reduces communication and computing costs associated with application and data encryption and decryption, making execution more efficient. In addition, applications with Intel® SGX support can be developed, integrated and executed on a specific Intel® processor-based platform, requiring only the installation of drivers and SDK adaptation, with no additional hardware or software environment, no changes to programming, and a lower learning curve. Meanwhile, the TEE hardware solutions based on Intel® SGX run more efficiently than federated learning implementations that adopt technologies such as secure multi-party computation and encryption.



Block insider attacks on applications to better protect code and data

Intel® SGX Application

The Intel® SGX application contains two parts, an untrusted startup component and a trusted component, with production code running in the "enclave" of the trusted component. Numerous solutions benefit from the additional protection provided by Intel® SGX, including Artificial Intelligence (AI) and Machine Learning (ML) processing, key management, proprietary algorithms, biometric protection, and more. Developers can support a distributed architecture by creating 1 to n enclaves that run collaboratively. While the program is running, Intel® SGX instructions create "enclaves" and execute them to specific areas of encrypted memory, and developers can restrict access to/from this area in order to prevent data breach.

Enclave authentication and data sealing

Intel® SGX supports local authentication or relying party remote authentication between enclaves to ensure that applications are not corrupted. One part of the application is loaded into an "enclave" for code and data measurement. The enclave report is then sent to the remote application owner's server, which in turn verifies that whether the enclave report is generated by a real Intel processor or not.

Data center authentication

Intel® SGX Data Center Authentication Source Code (Intel® SGX DCAP) allows enterprises, data centers and cloud service providers to build and deliver their own authentication services without the need for remote authentication from a third-party provider.

Enabling security model innovations

The fundamental function of Intel® SGX is to provide a higher level of isolation and authentication for the platform's Operating System (OS), application and hardware code, data, and core Internet Protocol (IP) to significantly reduce the likelihood of software attacks, and has been used to enhance the security of a wide range of use cases and applications, including:

- Key Management
- Blockchain
- Privacy-Enhancing Analytics & Workloads
- Applications at Runtime
- Hardware-Enhanced Content Protection
- Enhanced Application & Data Protection
- Edge Computing
- Digital wallet
- Communications & Messaging

Intel® SGX Resources

Intel® SGX Commercial License Information:

https://software.intel.com/sgx/request-license

Intel® SGX Software Development Kit (SDK):

https://software.intel.com/sgx/sdk

Download the Intel® SGX Documentation and Software Development Kit (SDK) at:

https://software.intel.com/sgx

Glossary of Terms Used in This Guidebook

Abbreviations	Full Name
ALS	Alternating Least Squares
API	Application Programming Interface
CART	Classification and Regression Tree
CNN	Convolutional Neural Networks
Data Mart	Data Mart
DDR SDRAM	Double Data Rate Synchronous Dynamic Random-Access Memory
Decision Tree	Decision Tree
DL	Deep Learning
DMLC	Distributed Machine Learning Community
GBDT	Gradient Boosting Decision Tree
GMF	Generalized Matrix Factorization
GRU	Gated Recurrent Unit
HDFS	Hadoop Distributed File System
HMM	Hidden Markov Model
IAaas	Intelligent Analysis as a service
Imbalance Ratio	Imbalance ratio
Intel DAAL	Intel Data Analytics Acceleration Library
Layer Fusion	Layer Fusion
LR	Logistics Regression
LSTM	Long Short-Term Memory
MF	Matrix Factorization
MKL	Math Kernel Library
MKL-DNN	Math Kernel Library for Deep Neural Networks
ML	Machine Learning
MLP	Multi-Layer Perception
NBM	Naive Bayesian Model
NCF	Neural Collaborative Filtering
NLP	Natural Language Processing
NNs	Neural Networks
NUMA	Non- Uniform Memory Access Architecture
One-Hot	One-Hot
POJO	Plain Ordinary Java Object
PR	Precision – Recall
RDD	Resilient Distributed Datasets
ResNet	Residual Net
RF	Random Forest
RNN	Recurrent Neural Network
RS	Recommender System
SQL	Structured Query Language
Streaming Model	Streaming Model
SVM	Support Vector Machine
WAD	Wide and Deep
XGBoost	eXtreme Gradient Boosting

Disclaimers:

Software and workload used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests such as SYSmark and MobileMark are measured using specific computer systems, components, software, operations, and functions. Any change to any of the aforementioned factors may cause test results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other product. For more information, go to www.intel.com/benchmarks.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit intel.com/benchmarks.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No component or product can be absolutely secure. Check with your system manufacturer or retailer, or learn more at intel.com.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations cover SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not make any warranties as to the existence, functionality or effectiveness of any optimizations on non-Intel microprocessors. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision: #20110804

No component or product can be absolutely secure.

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party data. You should review this content, consult other sources, and confirm whether referenced data is accurate.

To accelerate AI implementation, please visit:

Our official website
Intel.cn/ai

Micro Blog
@ Intel Business

WeChat
Intel Biz