2018 the 3rd IEEE International Conference on Cloud Computing and Big Data Analysis

Deep Learning Model Management for Coronary Heart Disease Early Warning Research

Yang Peili, Yin Xuezhen, Ye Jian, Yang Lingfeng
School of Computer Science and Software Engineering
East China Normal University
Shanghai, China
e-mail: yangpeili112@163.com

Zhao Hui
Shanghai Key Laboratory of Trustworthy Computing
Shanghai, China
e-mail: hzhao@sei.ecnu.edu.cn

Liang Jimin
School of Life Sciences and Technology
Xidian University
Xi'an, China

***Abstract*—Coronary Heart Disease (CHD) is one of the common diseases that threaten people's health and life. To facilitate the CHD early warning research, the deep learning based methods have drawn much attention. However, the literature mostly focuses on how to establish and optimize the CHD early warning models, while overlooking the *training data-model-experimental results* modeling lifecycle management. Aiming to promote the early warning research of CHD, we contribute a data management system integrated the CHD patient data with the deep learning model data. In the system, a deep learning model version tree is established to represent the relationship between the models. Tracking-Ancestors algorithm and Find-Specified-Ancestor algorithm are designed to conduct the lineage management of the deep learning model. Considering the big data characteristics of the patient data and deep learning model data, we compare the query response time and select MongoDB as the DBMS for the *Pdmdims* (*Patient Data & Deep Learning Model Data Integrated Management System*). The research results show that *Pdmdims* can provide an effective integrated data management platform for CHD early warning researchers.**

Keywords-deep learning model data; versioning mechanism; version lineage; CHD patient data; integrated management

I. INTRODUCTION

Coronary Heart Disease (CHD) is one of the most common diseases threatening people's health and life [1]. The traditional diagnosis method mainly relies on the doctors' professional knowledge and the clinic experience. In recent years, big data and deep learning based precision medical treatments have become a research hotspot [2] [3]. Deep Learning is a part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms [4]. The Deep Learning model, also called Deep Neural Network (DNN), is an artificial neural network with multiple hidden layers between the input layer and output layer [5]. To illustrate the Deep Learning model, Figure 1 shows an example of the network structure of LeNet5, a classical convolutional neural network. The input data are CT (computerized tomographic) slices of patients.

The process of establishing a Deep Learning model is usually complex and time-consuming. Figure 2 displays the deep learning modeling lifecycle for CHD early warning research. Commonly, some well-known models in similar task domains are selected as the reference models. Then given the training data and loss function, the training model task is conducted. If the accuracy is not satisfied, then the hyperparameters or network structure are adjusted until the accuracy is no longer improved.

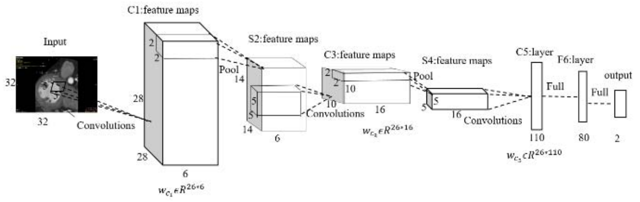


Figure 1. An Example of Deep Neural Network.

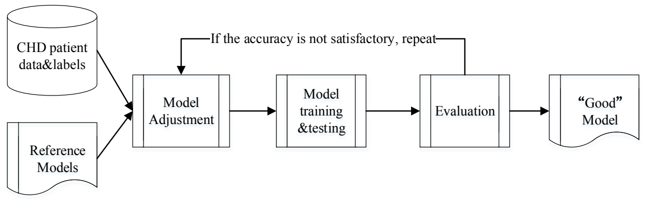


Figure 2. DNN modeling lifecycle for CHD early warning.

To date, most CHD early warning researchers focus on how to establish and optimize the early warning models, while ignoring the data management during the modeling lifecycle, such as the large number of training data, the models' network, and the accuracy. Each modeling experiment generates a particular *training data-model-experimental results* chain. The training data are obtained from the CHD patient data. File systems and spreadsheets are usually used to manage the data generated in the modeling lifecycle, which causes a rapid increase of the storage requests and inconvenience for researchers to compare and analyze the difference between experiments. Thus, there is an urgent need to supply CHD early-warning researchers a platform that integrates CHD early-warning

DNN model data and CHD patient data. When developing the platform, there are some challenges: (1) Patient data [6] involve a long-term follow-up progress of huge numbers of patients. The follow-up data include medical history, biochemical index, imaging examination and pathological examination, which have a significant big data feature. (2) Building a DNN model is a continuous optimization process, in which multiple models are generated based on the initial reference model. The relationship between those models needs to be established, so that the lineage of models can be traced. (3) Usually, it is technically necessary to compare the difference between the models in which each model contains the training data, hyper-parameter, DNN, weight values, and associated files. (4) During the model training stage, many *training data-training model-training results* data chains exist. If each data chain is stored separately, the training data are stored repeatedly and difficult to query.

The purpose of our research is to develop a deep learning model management system to overcome the challenges and satisfy the needs for the CHD early-warning researchers. The rest of the paper is organized as follows. In section 2, we briefly introduce the related work. In section 3, we display the conceptual schemas design of the DNN model data and CHD patient data. A DNN model version tree is established to represent the relationship of the *training data-training model-training results* chains. Tracking-Ancestors and Find-Specified-Ancestor algorithms are designed to manage the lineage of the DNN models. In section 4, we propose a framework called *Pdmdims*(Patient Data & Deep Learning Model Data Integrated Management System) which integrates the DNN model data and CHD patient data. The deep learning model and patient data query services are provided. In section 5, we convert the conceptual schema of patient data to relational and NoSQL logical schema. Experiment results guide us to select MongoDB to manage the data. The user interface to *Pdmdims* is displayed in section 6, and Section 7 concludes the paper.

II. Related Work

In recent years, medical data management has gained widely attention because of the rapid advancement of precision medical. Celesti et al. developed OIAS [7], a clinical data management system using column-family database Cassandra and optimized Cassandra based on Health Level Seven Standard. Bahga A et. al. [8] proposed an interactive Electronic Health Record management system based on HBase. Mazurek M. [9] combined the relational database and NoSQL database to manage the patient clinic data. The NoSQL database is used to store the unstructured data like CT images, and the relational database is used to store the features extracted from the clinic records. Data Café [10] is a data warehousing platform that integrates the research data based on biomedical data from disparate and multi-modality data sources. Machine Learning method is widely used in healthcare and great progress has achieved. Most researchers focus on model training and optimization [11] [12]. More data management challenges arise in machine learning, especially in deep learning workloads [13]. PDBMS [14] incorporates a predictive model management component to support the machine learning models management. The model managers choose the models and the data managers provide the data according to the predictable task. ModelDB [15] is an end-to-end system to manage the machine learning models. It can automatically capture the model information from the experiment environment logs by invocating the local machine learning library. The model related data are stored in relational database. DataHub [16] is a data set version management system. A data saving strategy is proposed by recoding the deltas between two versions that minimized the storage cost. Hui Mao et al [17] indicated the data version of deep learning modeling lifecycle and implemented a management system called ModelHub. A read-optimized parameter system PAS is proposed to manage the float parameters of the checkpoint in DNN model training phases.

III. Conceptual Schema Design

A. CHD Patient Data Conceptual Schema Design

1) CHD patient data introduction

The typical information of CHD patient medical data are summarized as Basic Information, Medical History, Medicine Information, Check Information, Vital Sign, Diagnostic Results, Special Check-up, Plaque Information, Exercise Information, Unhealthy Habits and Family History. These data are the basic information of CHD patient that provides the CHD risk factors.

2) Patient conceptual schema design

Considering the characteristics of the CHD patient data, we design the conceptual schema of the patient data abstracted from patient data summarization. Respectively, the entities are Patient, CheckInfo, DiagnositicResults, MedicalHistory, MedicineInfo, VitalSign, ExerciseInfo, UnhealthyHabits, PlaqueInfo, FamilyHistory and CheckItem. Figure 3 shows the details of the conceptual data model. Since patient data involve a long-term follow-up, each patient is subjected to multiple checks over 3 to 5 years. Thus, the relationship between the Patient and CheckInfo entity is one-to-many. One patient may have multiple medical histories, different exercise habits, unhealthy life habits, and family disease history, thus the relationship between Patient entity and MedicalHistory entity, ExerciseInfo entity, UnhealthyHabits entity and FamilyHistory entity is one-to-many respectively. The doctor prescribes medicines to patient for one disease, so the relationship between MedicalHistory and MedicineInfo entity is one to many. Check-up information includes vital sign information and some special medical checks, thus the relationship between the CheckInfo entity and VitalSign entity is one-to-one, and the relationship between the CheckInfo and CheckItem is one-to-many. Normally, doctors make a diagnosis based on the patient check information, so the CheckInfo entity is a one-to-one relationship with the DiagnosticResult entity. The diagnostic results information may contain multiple plaques information, so the relationship between DiagnosticResult entity and PlaqueInfo entity is one-to-many.

B. DNN Model Data for CHD Early Warning

1) DNN Model Data

The DNN model data for CHD early warning can be represented as model name, input data, DNN definition, hyperparameters, weight parameters and the files associated with a model, such as initial configurations. Building a DNN model for a specific task is a progress of continuous optimization after specifying the input training data and output loss functions. To get a more accurate result, researchers always repeat the experiment by tuning the hyperparameters, adjust the DNN network structure or change the input data. During the model training stage, multiple "training sample-training model-training result" data chains are generated.

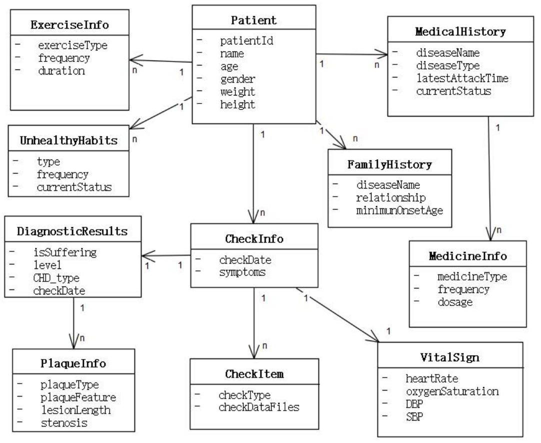


Figure 3. Conceptual schema of CHD patient data.

2) DNN Model Version

To manage multiple *training sample-training model-training result*s data chains, we propose a model version strategy. The version number is used to identify the model. Thus, we give the version number definition first which represents the version identity and the relationship among the DNN models.

Definition: The *ith* version number of the model is defined as V_i. $V_i = M_i . P_i . U_i . C_i$, while M_i and C_i both are decimal number, U_i is a binary number with N bits. N is the number of the factors that determine the generation of a new version. Each factor is represented using a bit. V_{i-1} is the parent version of V_i. $P_i = P_{i-1}.U_{i-1}.C_{i-1}$, $M_i = M_{i-1} + 1$, $M_i \in N^*, C_i \in N^*$, while N^* is positive integer set. $V_0 = 1.0.000.1$. We use "." to label the four distinct parts of the version number.

According to the model version definition and the relationship between the versions of the model, we establish a DNN model version tree. For the example of Figure 4, the version number of the DNN models is made up four parts which are the major version number, parent version number, update information, and the count of the same type adjustment based on the parent model. A new version is generated when the hyperparameters are changed or the definition of the network, the input training data are changed. The major version number increases in order. The parent version number records the relationship of the ancestor model. Three bits are used to represent the modification type indicates whether the hyperparameter, network, or input data are changed. As the grey color filled circles showed in Figure 4, the root is the initial version of the model that the version number is assigned as 1.0.000.1. The new version is generated when the hyperparameters are tuned. The new version number 2.0.000.1.100.1 is obtained. Similarly, another new version number 3.0.000.1.100.1.010.2 is generated when the model's network structure is changed.

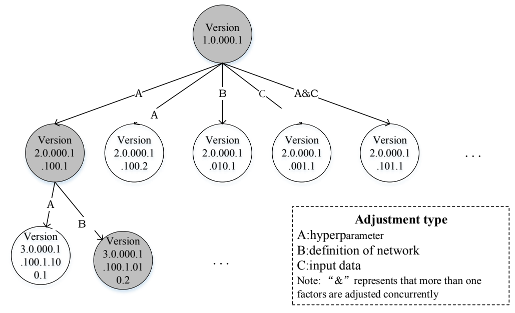


Figure 4. An example of a DNN model version tree.

3) Algorithms for DNN Data Model Lineage Management

Algorithm 1. Tracking-Ancestor

```
Function trackAncestor(version,modelName)
Input:  version        //current version number
        modelName  //name of the DNN model
    Output:  versionAncestorList
             //the list of the current version's ancestor versions.

currentVersion <- version;
A[ ] <- split(version, splitFlag);
  ▷split the parent version number by the flag
currentMainNum <- A[0];          // get the major number.
IF currentMainNum>1 THEN
  parentMainNum <- currentMainNum-1;
  j <- A.length-3          //pointer j points to the bottom third of A.
  parentVersion <- formatPartentVersion(parentMainNum, A[1…j]);
    // uniformed the version number
  parentVersionModel <- findParentModel(parentVersion,modelName)
  versionAncestorList.add(parentVersionModel)
  trackAncestor(parantVersion, modelName)
    //find the information of the ancestors recursively
END IF
return versionAncestorList
```

Algorithm 2. Find-Specified-Ancestor

```
Function findSpecifiedAncestor(version,modelName ,n)
Input: version              //the current version number
       modelName        //name of the DNN model
       n //find the nth ancestor.
         //we defined the parent version is the first generation ancerstor
  Output: specifiedAncestorVersion

currentVersionversion;
A[ ] <- split(version, sliptFlag);
  // split the parent version number by the flag
currentMainNum <- A[0];  // get the major number.
ancestorMainNum <- currentMainNum-n;
IF ancestorMainNum>=1 THEN
  j<-(A.length-1)-2*n
  specifiedAncestorVersion<-
  formatSpecifiedAncestorVersion (ancestorMainNum, A[1…j])
ELSE
    return ERROR
END IF
specifiedVersionAncestor <- findParentModel(version,modelName)
return  specifiedVersionAncestor
```

In this section, two algorithms for DNN data model lineage management are proposed. Tracking-Ancestors is to track the ancestor of the current version. Find-Specified-Ancestor is to query the specific DNN model.

IV. The Architecture of *Pdmdims*

To promote the research of CHD early warning research, we develop a system called *Patient Data & Deep Learning Model Data Integrated Management System* (*Pdmdims*) to manage the deep learning lifecycle data during the continuous optimization process. The CHD patient data are integrated in this system to provide the training data to the CHD early warning model building. Furthermore, the system provides some cohort analysis services.

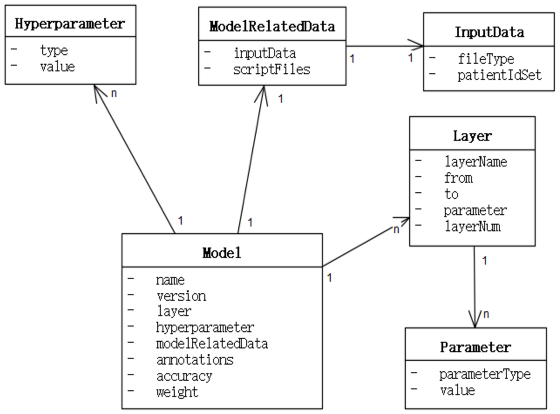


Figure 5. Conceptual schema of DNN model.

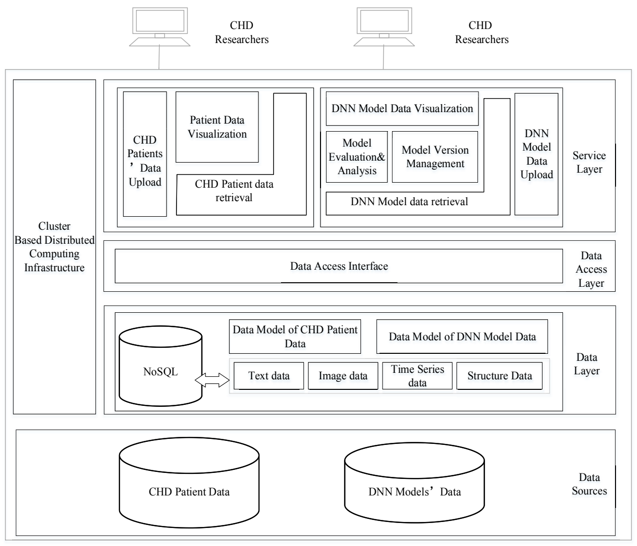


Figure 6. The architecture of *Pdmdims*.

Pdmdims is designed as a hierarchy structure. It has three layers which are Data Layer, Data Access Interface Layer and User Service Layer from bottom to top. The Data Layer focuses on management of the patient data and DNN model data. Data Access Interface Layer provides the interface to the database and the encapsulation of the database manipulation. This layer is responsible for decreasing the complexity of the database access and improving the query efficiency. Service Layer provides data query service, data import service, and visualization services. The architecture of *Pdmdims is displayed in* Figure 6.

V. Logical Schema Design

In this section, we convert the conceptual schema of patient data to two types of logical schema according to the data characteristics. The one is a relational schema based on MySQL and the other is a document-oriented schema based MongoDB. The query performances are compared to select a best DBMS.

1) The Logical Schema of the Patient Data Based on MySQL

According to the data conceptual schema design, we convert the scheme showed in Figure 3 into eleven relational tables. Each entity is transferred to a table. For the one-to-many relationship between entities, we use the foreign key constraint. Figure 7 displays the schema relation. Unstructured data like CT image is stored in the file system while the address is stored in the table.

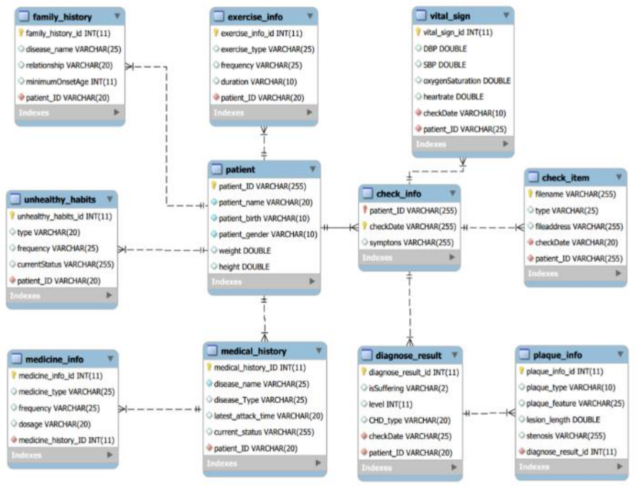


Figure 7. The Logical Schema Design based on MySQL.

2) The Logical Schema Design Based on MongoDB

MongoDB is a document-oriented NoSQL database. The data of each patient is organized as one document. The MedicalHistory, ExerciseInformation, UnhealthyHabits, FamilyHistory and the CheckInfo are organized as the nested sub-document. MedicineInfo records the medicine that patients have taken, so MedicineInfo is organized as a nested sub-document of MedicalHistory. Similarly, the PlaquesInfo is part of DiagnositicResults. Thus PlaquesInfo is organized as a nested sub-document of DiagnositicResults. All the attributes in the conceptual schema of each entity are transferred as keys in a document structure. Unstructured data like CT images is stored in the GridFS. The design details are displayed in Figure 8.

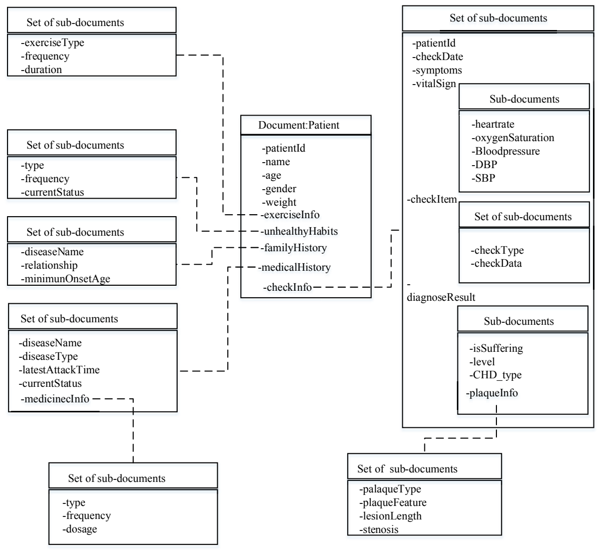


Figure 8. Logical schema design based on MongoDB.

B. *The Database Selection Experiment Based on Patient t Data*

The experiment is conducted on a PC cluster composed of 4 PC computers. Table I presents the configuration of the cluster.

TABLE I. CONFIGURATION OF THE EXPERIMENT ENVIRONMENT

Category	**Value**
Memory	8GB
CPU	3.2Hz
HardDisk	1T
Operating system	CentOS6.8
MySQL	MySQL7.4.14
MongoDB	MongoDB3.29
HBase	HBase1.2.3

We conduct the experiments using 5000 simulated patients' data with three follow-up years, which are generated based on the real patient clinic data. Query performances of four typical queries based different logical schema are evaluated to find the optimal schema design. For examples:

- Query 1: Find out the diastolic pressure of patient F378495 who had a medical check on Oct 4th 2015.
- Query 2: Check the systolic pressure changes of patient F378495 from June 1st 2014 to Dec 23rd 2016.
- Query 3: Find out all the check data of patients who had medical check in 2014 and were born in 60's and has diabetes.
- Query 4: Get all CT slices of female patients whose heart rate is in [80,110] and who have already been diagnosed coronary heart disease and had medical check between Jan. 2013 to Mar. 2013.

When using relational DBMS to organize the CHD patient data, join operation is needed to finish the complex query. However, we organize all the data of one patient into a document, which is stored continuously on the disk. In addition, the patient data contains many small files. For example, CT image data usually consist of 400-600 CT slices. Each slice is about 500KB big. Figure 10 displays the query time results. Q1 and Q2 are about structured data query. Q3 and Q4 are for the unstructured data query. All are shown that MySQL based query time is longer than MongoDB based. Moreover, MongoDB has better scalability. Thus, MongoDB is selected for *Pdmdims*.

C. *The Logical Schema Design of DNN Model Data Based on MongoDB*

Because the MongoDB based logical schema design based on CHD patient data has better query performance, we choose MongoDB to manage both CHD patient data and DNN model data for *Cdmdims*. Based on the design of the DNN model data conceptual schema, each version of a DNN model is organized in a document. All the attributes in the DNN model conceptual schema are transferred as keys. Hyperparameter, DNN structure and model related data of one model version are organized in the nested sub-document. Figure 11 presents the details of the logical schema.

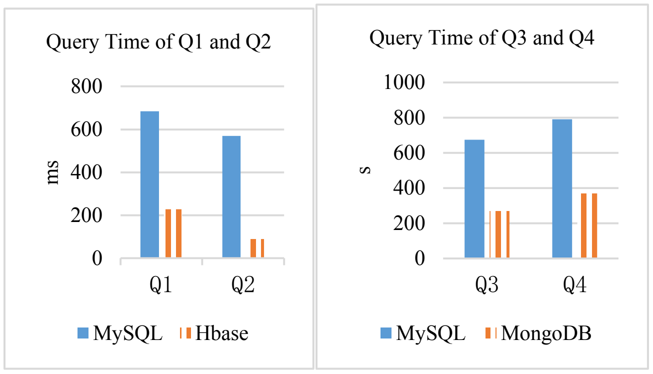


Figure 9. The query time based on two different DBMS.

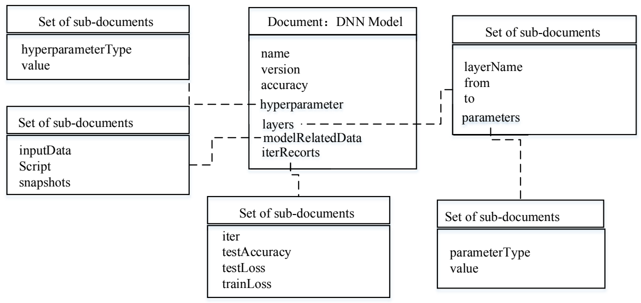
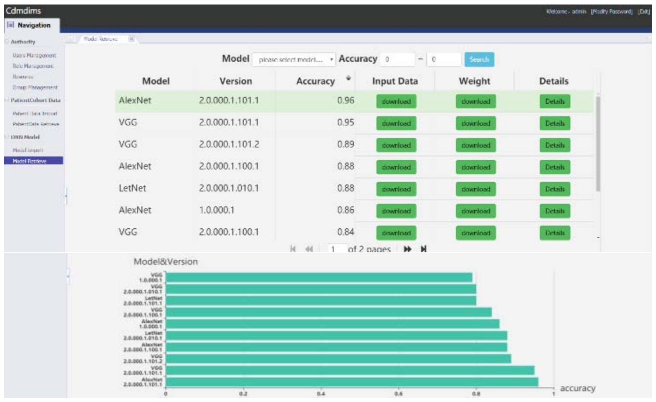


Figure 10. Logical Schema of DNN Model Data Based on MongoDB.

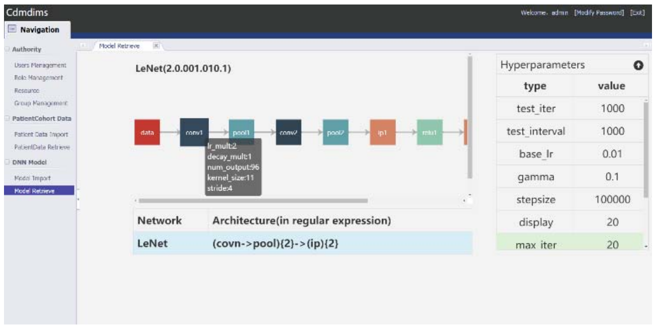
VI. USER INTERFACE TO *PDMDIMS*

The interactive user interface is developed in *Pdmdims* for exploring, querying the patient data, comparing, and analyzing the DNN model data. A low latency distributed mass data interactive query engine called Apache Drill is used to improve the query efficiency. A model import interface is provided for medical researchers to upload the model related data including the training data, network definition, hyperparameter, scripts, and the weight to generate a new version of a DNN model. Figure 11 depicts the DNN model query interface which displays the model information with Top10 accuracy. Figure 12 visually illustrates the network structure of a DNN model and the parameters related information of each layer in DNN.



Model	Version	Accuracy	Input Data	Weight	Details
AlexNet	2.0.000.1.101.1	0.96	download	download	Details
VGG	2.0.000.1.101.1	0.95	download	download	Details
VGG	2.0.000.1.101.2	0.89	download	download	Details
AlexNet	2.0.000.1.100.1	0.88	download	download	Details
LetNet	2.0.000.1.010.1	0.88	download	download	Details
AlexNet	1.0.000.1	0.86	download	download	Details
VGG	2.0.000.1.100.1	0.84	download	download	Details

Figure 11. The DNN model information with Top 10 accuracy.



LeNet(2.0.001.010.1)

lr_mult:2
decay_mult:1
num_output:96
kernel_size:11
stride:4

Network	Architecture(in regular expression)
LeNet	(covn->pool){2}->(ip){2}

Hyperparameters

type	value
test_iter	1000
test_interval	1000
base_lr	0.01
gamma	0.1
stepsize	100000
display	20
max_iter	20

Figure 12. The Network Structure of a DNN model and Hyperparameter Display.

VII. CONCLUSION

CHD is a common chronic disease threatening people's health and life. The CHD early warning research has benefited from the deep learning techniques and big patient data. The Deep learning based precision medical has attracted the researcher's extensive attention while the more and more data management challenges arise in deep learning workloads. To promote the research of CHD early warning, we have explored the key challenges about the data management concerning the CHD early warning research based on the deep learning method and developed a data management system *Pdmdims* which integrates the CHD patient data and the DNN model data. The Patient data are used not only as the training data of the CHD early warning model, but also as the data for cohort analysis. In *Pdmdims* the DNN model version tree is established to show the relationship between the different model versions. The Tracking-Ancestors and the Find-Specified-Ancestor algorithms are designed to conduct the lineage management of the deep learning model. The conceptual data schema about the patient data and the model data are designed and transferred to one relational schema and document-oriented NoSQL schemas. The query experiment based on CHD patient data guides us to choose the document-oriented NoSQL database MongoDB as the database for *Pdmdims*. The user interfaces of *Pdmdims* are presented to demonstrate that the researchers can explore and analyze the CHD patient data and the DNN model data visually and effectively.

ACKNOWLEDGMENT

This work was supported by grants from National Key R&D Program of China (2016YFC1300300).

REFERENCES

[1] Wang E Y, Dixson J, Schiller N B, et al. Causes and Predictors of Death in Patients With Coronary Heart Disease (from the Heart and Soul Study)[J]. American Journal of Cardiology, 2017, 119(1):27.

[2] Dinggang Shen, Guorong Wu, HeungIl Suk. Deep Learning in Medical Image Analysis[J]. Annual review of biomedical engineering, 2017, 19:221.

[3] Litjens G, Kooi T, Bejnordi B E, et al. A survey on deep learning in medical image analysis.[J]. Medical Image Analysis, 2017, 42(9):60.

[4] Lecun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553):436-444.

[5] Schmidhuber J. Deep Learning in neural networks: An overview.[J]. Neural Networks, 2014, 61:85-117.

[6] Arriola L, MartinezCamblor P, Larrañaga N, et al. Alcohol intake and the risk of coronary heart disease in the Spanish EPIC cohort study.[J]. Heart (British Cardiac Society), 2010, 96(2):124.

[7] Celesti A, Maria F, Romano A, et al. An OAIS-based Hospital Information System on the Cloud: Analysis of a NoSQL Column-Oriented Approach[J]. IEEE Journal of Biomedical & Health Informatics, 2017, PP(99):1-1.

[8] Bahga A, Madisetti V K. Healthcare Data Integration and Informatics in the Cloud[J]. Computer, 2015, 48(2):50-57.

[9] Mazurek M. Applying NoSQL Databases for Operationalizing Clinical Data Mining Models[J]. Communications in Computer and Information Science, 2014, 424(1):527-536.

[10] Kathiravelu P, Sharma A. A Dynamic Data Warehousing Platform for Creating and Accessing Biomedical Data Lakes[C]// International Workshop on Data Management and Analytics for Medicine and Healthcare. Springer-Verlag New York, Inc. 2016:101-120.

[11] Suzuki S, Shouno H. Support Vector Machine Histogram: New Analysis and Architecture Design Method of Deep Convolutional Neural Network[J]. Neural Processing Letters, 2017(4):1-16.

[12] Wang W, Chen G, Dinh A T T, et al. SINGA: Putting Deep Learning in the Hands of Multimedia Users[C]// ACM International Conference on Multimedia. ACM, 2015:25-34.

[13] Kumar A, Boehm M, Yang J. Data Management in Machine Learning: Challenges, Techniques, and Systems[C]// ACM International Conference on Management of Data. ACM, 2017:1717-1722.

[14] Akdere M, Çetintemel U, Riondato M, et al. The Case for Predictive Database Systems: Opportunities and Challenges[C]// CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011167-174.

[15] Vartak M, Subramanyam H, Lee W E, et al. ModelDB:a system for machine learning model management[C]// The Workshop on Human-In-The-Loop Data Analytics. ACM, 2016:1-3.

[16] Bhardwaj A, Bhattacherjee S, Chavan A, et al. DataHub: Collaborative Data Science & Dataset Version Management at Scale[J]. Computer Science, 2014.

[17] Miao H, Li A, Davis L S, et al. Towards Unified Data and Lifecycle Management for Deep Learning[C]// IEEE, International Conference on Data Engineering. IEEE, 2017:571-582.