

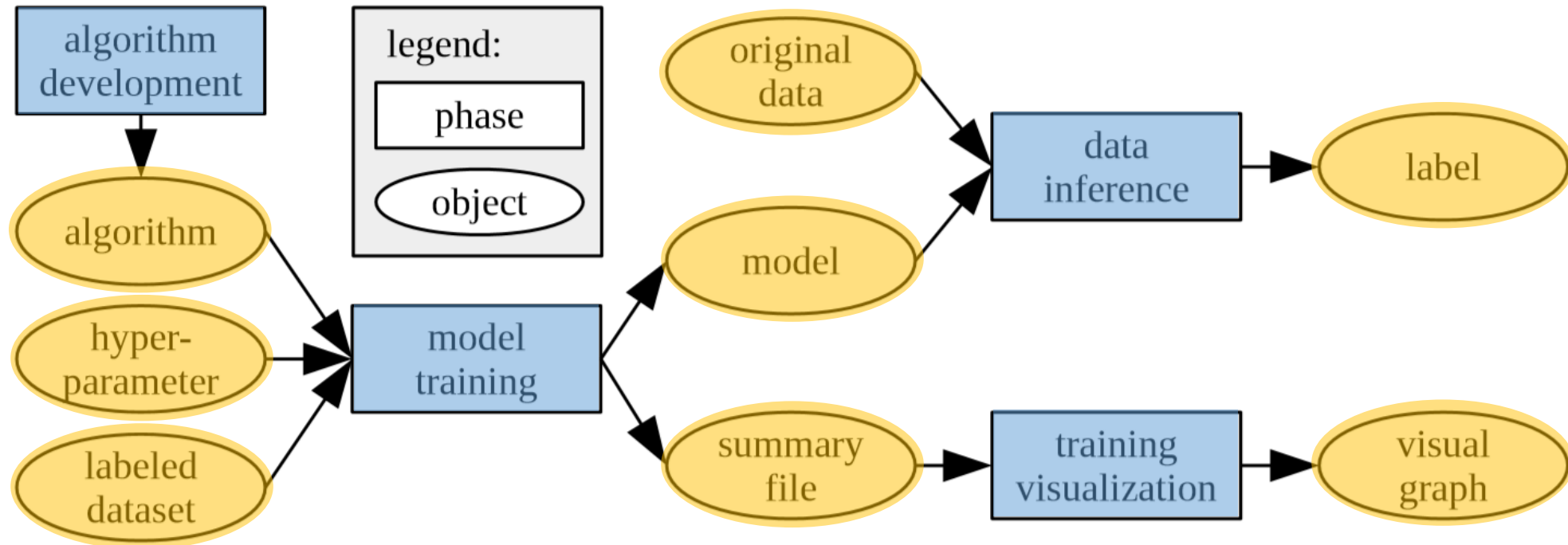
# OMProv: Provenance Mechanism for Objects in Deep Learning

Jian Lin, Dongming Xie  
Oriental Mind, China

# Contents

- Background
- Research problems
- Solution: OMProv
  - Version graph abstraction
  - Version inference algorithm
  - Optimizations
- Implementation
  - Version provenance in OMAI
  - Application practice
- Conclusion

# Background: workflow of deep learning



# Research problems

- Related work
  - Graph-based provenance methods: Open Provenance Model, Acar et al. 2011, ...
  - Provenance methods for datasets: DataLab, MLdp, ...
  - Provenance methods for models: ModelDB, ModelHub, ...
  - Provenance methods for whole ML/DL lifecycles: ProvDB, ...
  - Visualized ML/DL provenance tools: MEX, Runway, ...
- Problems and expectations

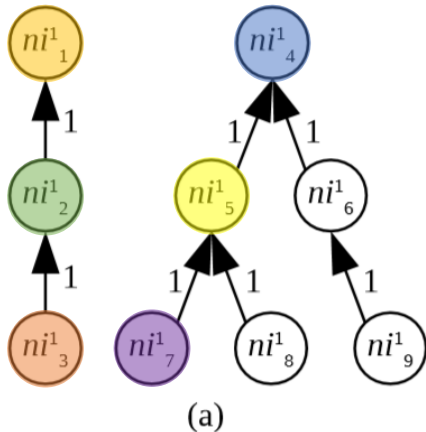
Existing	Expected
Qualitative version relationship	Quantitative version relationship
User-specified version relationship for output objects	Automatically managed version relationship for output objects
VCS-style command-line or web-based auxiliary tools	Native mechanism integrated with cloud services

# Version graph abstraction

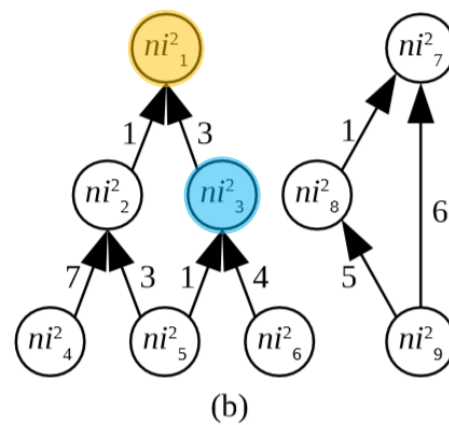
- OMProv

- Basis: weighted directed acyclic graph (wDAG)
- Core idea: for an output object, a new version  $i$  inherits an old version  $j$ , if and only if each provenance version that creates version  $i$  inherits the corresponding provenance version that creates version  $j$  respectively.

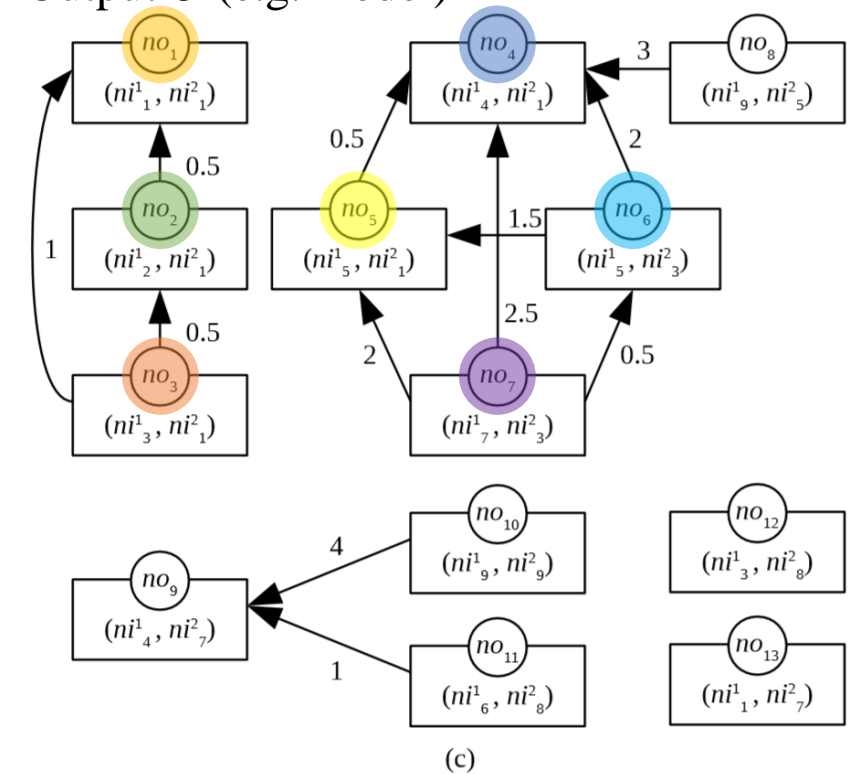
Input  $I_1$  (e.g. algorithm)



Input  $I_2$  (e.g. dataset)



Output  $O$  (e.g. model)



$I_1$  (algorithm) +  $I_2$  (dataset) model training  $O$  (model)

# Quantitative version relationship

- Weight on an edge: the amount of difference between versions
- Weight function: to accumulate and average the amounts of difference between the source and destination versions in all provenance objects

## Input objects

$$W(ei_{i,j}^p)$$

`wdiff -s` — for text files

`diff -r` — for any binary datasets

## Output objects

$$W(eo_{i,j}) = \text{weight}(S_{RI_{i,j}}) = \frac{1}{n} \sum_{p=1}^n W(ril_{i,j}^p)$$

$\text{weight}(\cdot)$  — weight function

$RI_{i,j}^p$  — provenance route set of input object  $p$

$S_{RI_{i,j}}$  — set of all provenance route sets

$ril_{i,j}^p$  — the lightest provenance route of input object  $p$

# Version inference algorithm

---

## Algorithm 1: Version inference algorithm

---

```

1 add a node  $no_i$  to  $nodes(GO)$ 
2 for each  $no_j$  in  $nodes(GO) \setminus \{no_i\}$  do
3    $flag_{reach} \leftarrow true$ 
4    $S_{RI_{i,j}} \leftarrow \emptyset$ 
5   for each  $ni_k^p$  in  $ivtuple(no_i)$  do
6     if  $reach(GI^p, ni_k^p, ivtuple(no_j)[p]) = false$  then
7        $flag_{reach} \leftarrow false$ 
8       break
9     else
10      add a set  $routes(GI^p, ni_k^p, ivtuple(no_j)[p])$  to  $S_{RI_{i,j}}$ 
11    end
12  end
13  if  $flag_{reach} = true$  then
14    add a directed edge  $eo_{i,j}$  (from  $no_i$  to  $no_j$ ) to  $edges(GO)$ 
15     $W(eo_{i,j}) \leftarrow weight(S_{RI_{i,j}})$ 
16  end
17 end

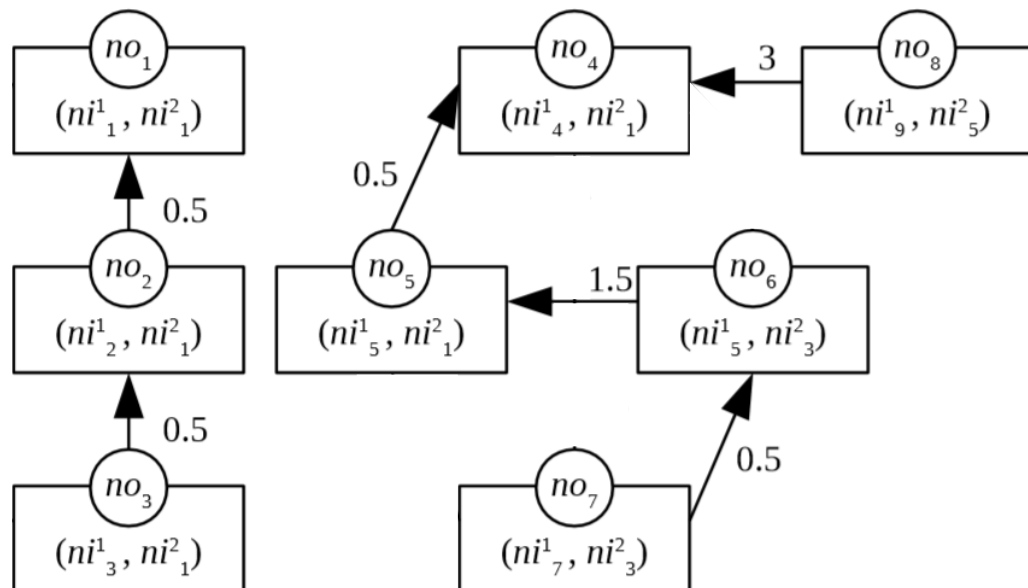
```

---

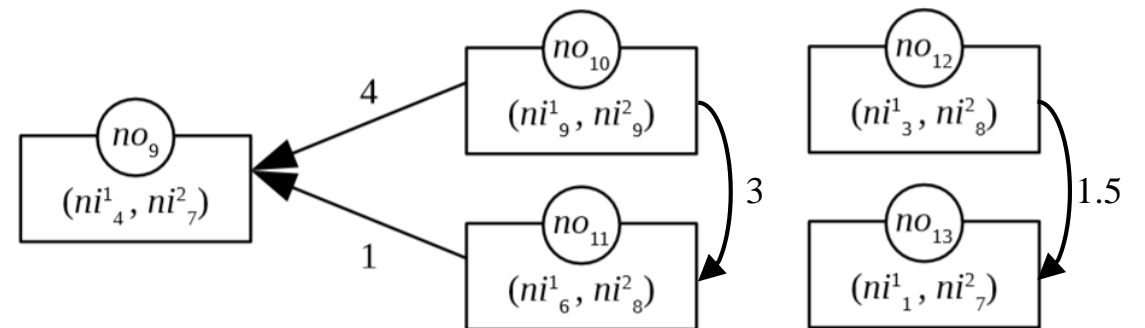
- Time analysis
  - Kinds of input objects:  $n$
  - Amount of versions of a certain output object:  $y$
  - Execution time of Algorithm 1:  
 $t_{total} = y \cdot (n \cdot t_{check} + t_{update})$
- Complexity
  - For  $GI^p$  (input version graphs), classic transitive closure algorithm:  $O(1)$  for reachability checking
  - For  $GO$  (output version graph), simple graph without extra indexes:  $O(1)$  for node/edge updating

# Optimizations

- Redundant edge avoidance



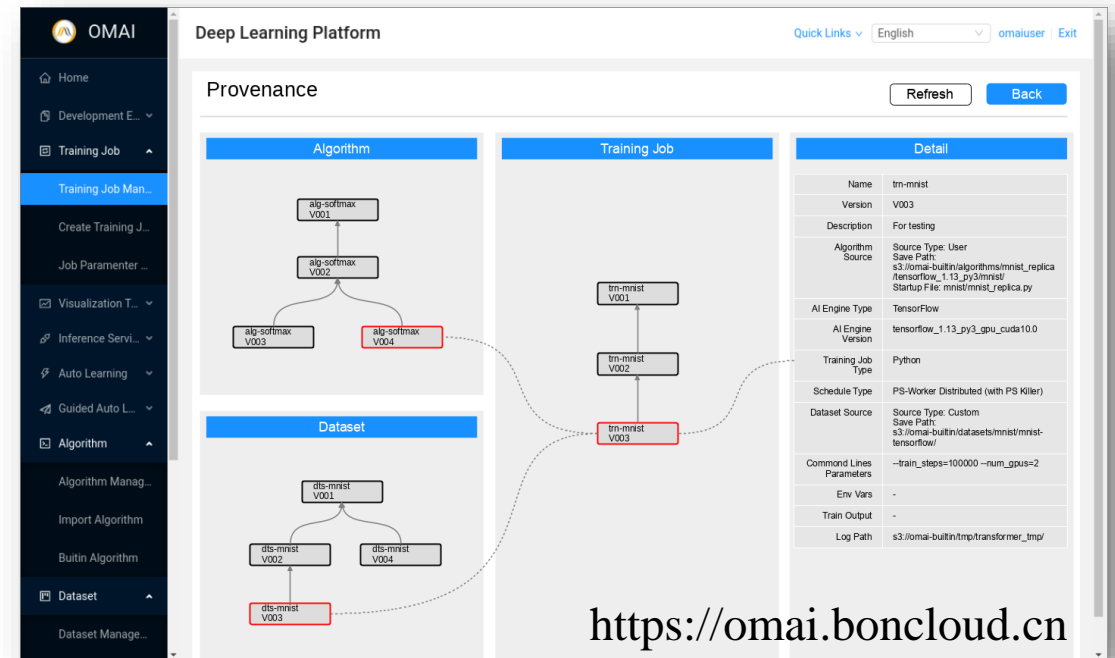
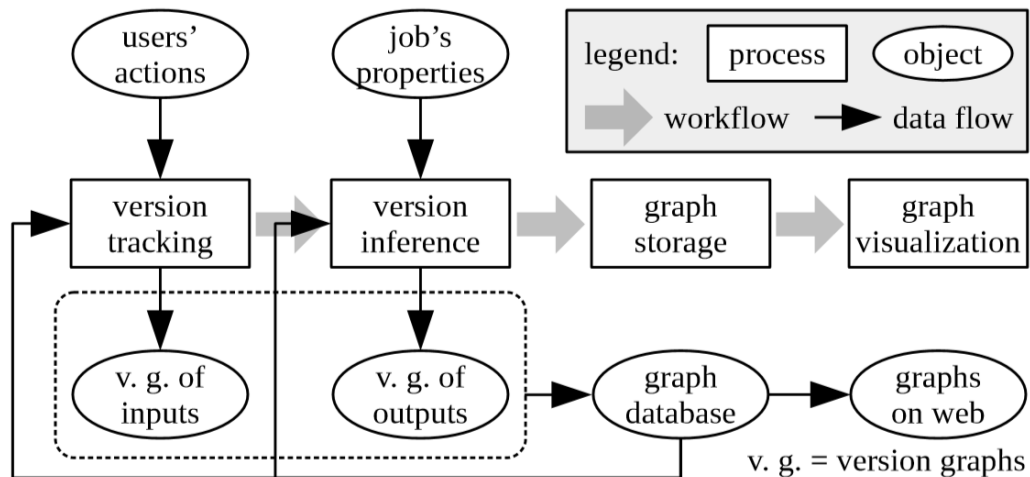
- Reverse version inference

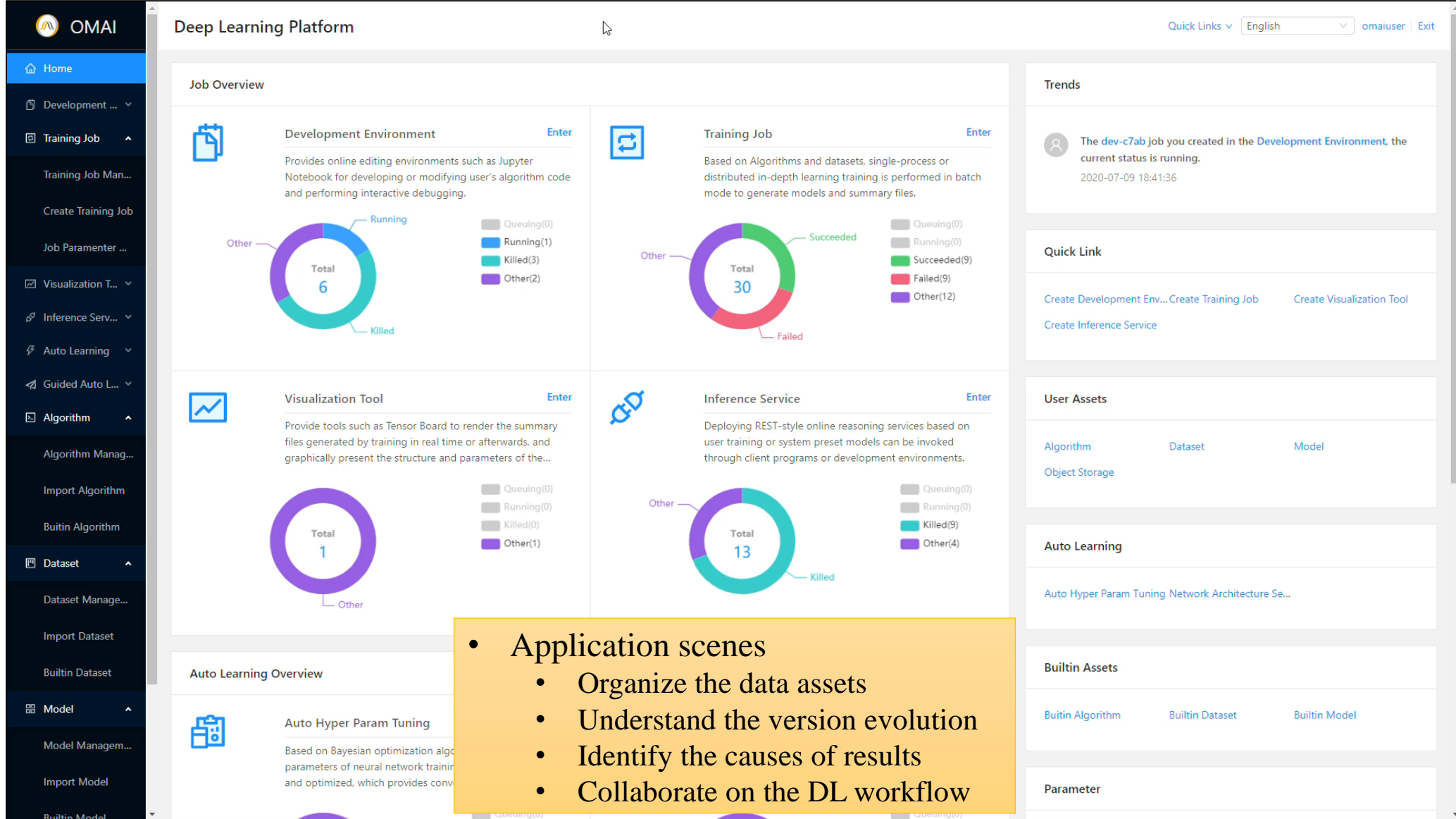




# Implementation

- OMAI Deep Learning Platform
- Graph database: ArangoDB





Deep Learning Platform

Quick Links English omaiuser Exit

### Job Overview

#### Development Environment

Provides online editing environments such as Jupyter Notebook for developing or modifying user's algorithm code and performing interactive debugging.

Total 6

- Queuing(0)
- Running(1)
- Killed(3)
- Other(2)

#### Training Job

Based on Algorithms and datasets, single-process or distributed in-depth learning training is performed in batch mode to generate models and summary files.

Total 30

- Queuing(0)
- Running(0)
- Succeeded(9)
- Failed(9)
- Other(12)

#### Visualization Tool

Provide tools such as Tensor Board to render the summary files generated by training in real time or afterwards, and graphically present the structure and parameters of the...

Total 1

- Queuing(0)
- Running(0)
- Killed(0)
- Other(1)

#### Inference Service

Deploying REST-style online reasoning services based on user training or system preset models can be invoked through client programs or development environments.

Total 13

- Queuing(0)
- Running(0)
- Killed(9)
- Other(4)

### Auto Learning Overview

#### Auto Hyper Param Tuning

Based on Bayesian optimization algorithm, parameters of neural network training are automatically searched and optimized, which provides convenient and efficient parameter tuning.

### Trends

The dev-c7ab job you created in the Development Environment, the current status is running.  
2020-07-09 18:41:36

### Quick Link

Create Development Env... Create Training Job Create Visualization Tool  
Create Inference Service

### User Assets

Algorithm Dataset Model  
Object Storage

### Auto Learning

Auto Hyper Param Tuning Network Architecture Se...

### Builtin Assets

Builtin Algorithm Builtin Dataset Builtin Model

### Parameter

- Application scenes
  - Organize the data assets
  - Understand the version evolution
  - Identify the causes of results
  - Collaborate on the DL workflow

# Conclusion

- Contributions
  - A wDAG-based version graph abstraction and a version inference algorithm
  - The provenance mechanism integrated in the OMAI deep learning platform
- Future work
  - Application cases studies on analyzing algorithm issues, improving model performance, and achieving model reproducibility
  - Integrate with the version control system for code to record the versions of algorithms automatically
  - Integrate with model inference services to manage the versions of online services easily

Thanks!