OMProv: Provenance Mechanism for Objects in Deep Learning

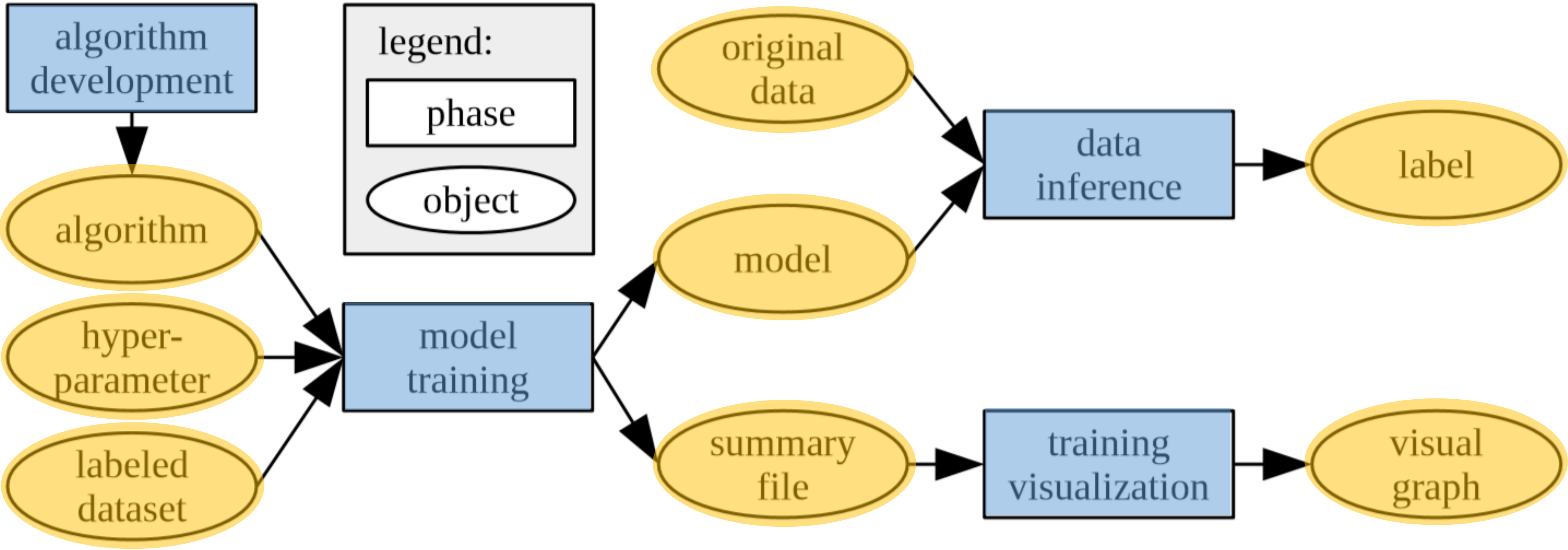
Jian Lin, Dongming Xie

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

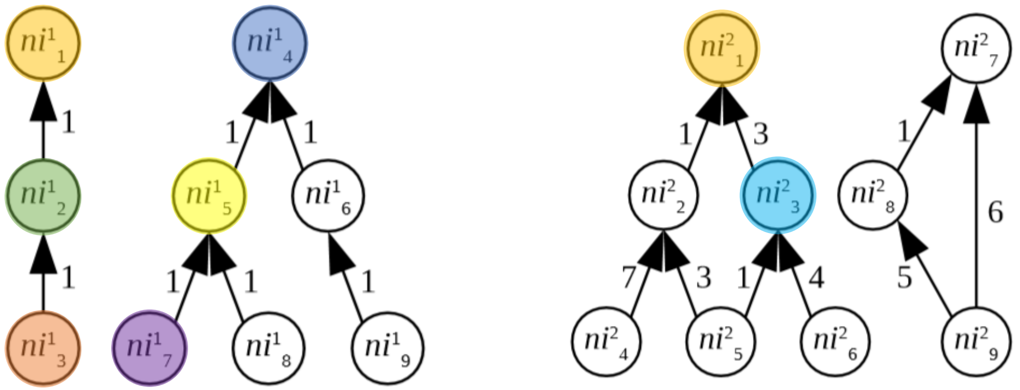
Existing	Expected
Qualitative version relationship	Quantitative version relationship
User-specified version relationship for output objects	Automatically managed version relationship for output objects
VCS-style command-line or web-based auxiliary tools	Native mechanism integrated with cloud services

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.

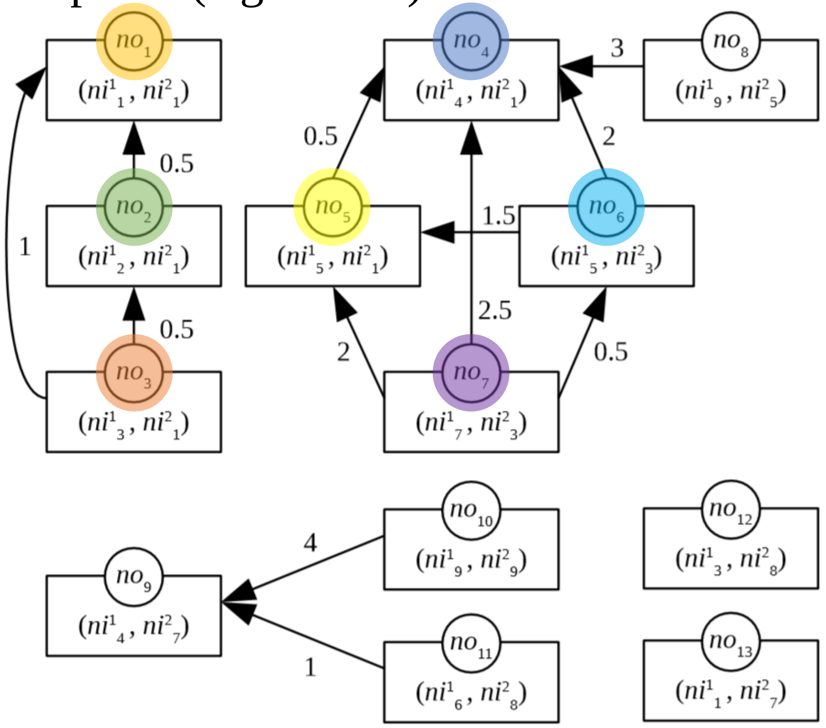
Input I_1 (e.g. algorithm)

Input I_2 (e.g. dataset)



(a) (b)

Output O (e.g. model)



(c)

I_1 (algorithm) + I_2 (dataset) $\xrightarrow{\text{model training}}$ 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	Output objects
$W(ei_{i,j}^{p})$ `wdiff -s` — for text files `diff -r` — for any binary datasets	$W(eo_{i,j}) = weight\left(S_{RI_{i,j}}\right) = \frac{1}{n}\sum_{p=1}^{n} W\left(ril_{i,j}^{p}\right)$ $weight(\cdot)$ — 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

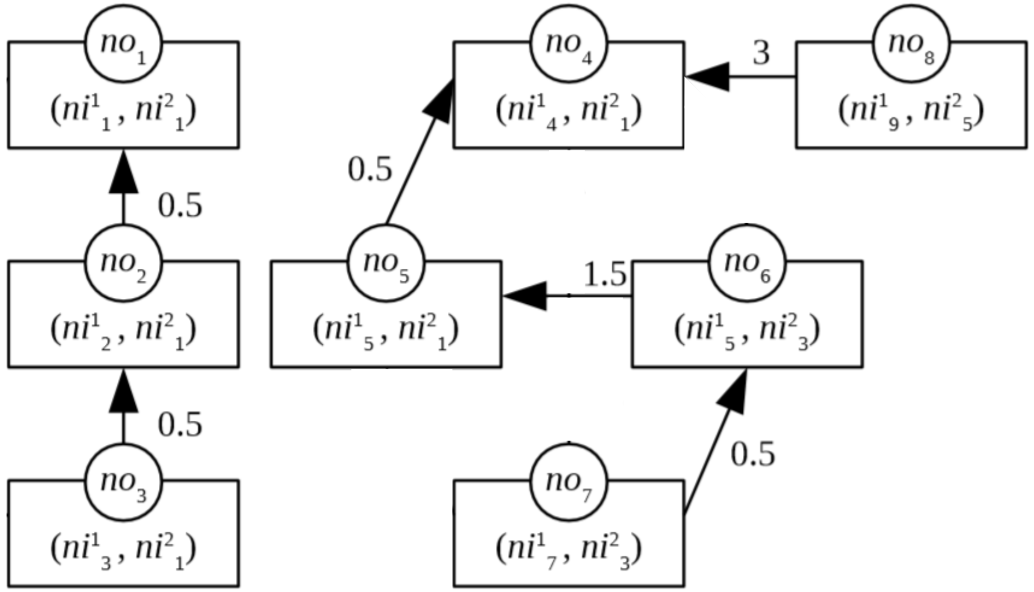
Algorithm 1: Version inference algorithm

add a node no_i to $nodes(GO)$
for each no_j in $nodes(GO) \setminus \{no_i\}$ **do**
  $flag_{reach} \leftarrow true$
  $\mathbf{S}_{RI_{i,j}} \leftarrow \emptyset$
  for each ni_k^p in $ivtuple(no_i)$ **do**
    if $reach(GI^p, ni_k^p, ivtuple(no_j)[p]) = false$ **then**
      $flag_{reach} \leftarrow false$
      break
    else
      add a set $routes(GI^p, ni_k^p, ivtuple(no_j)[p])$ to $\mathbf{S}_{RI_{i,j}}$
    end
  end
  if $flag_{reach} = true$ **then**
    add a directed edge $eo_{i,j}$ (from no_i to no_j) to $edges(GO)$
    $W(eo_{i,j}) \leftarrow weight(\mathbf{S}_{RI_{i,j}})$
  end
end

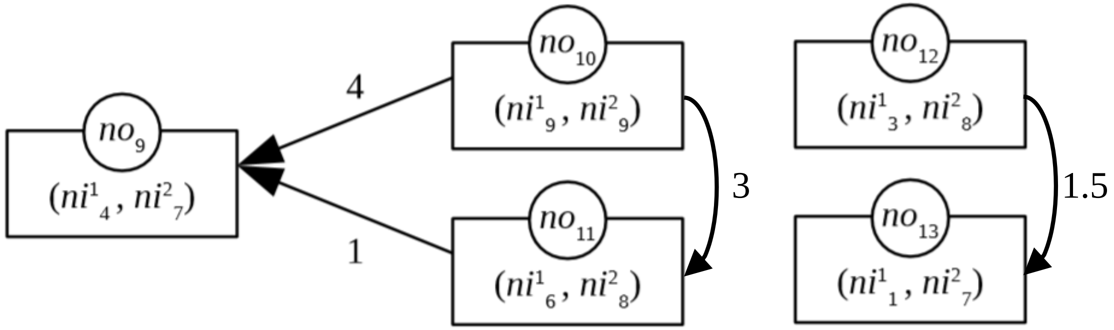
- Time analysis
  - Kinds of input objects: n
  - Amount of versions of a certain output object: y
  - Execution time of Algorithm 1: $t_{total}=y\cdot(n\cdot t_{check}+t_{update})$
- Complexity
  - For GI^p (input version graphs), classic transitive closure algorithm: $O(1)$ for reachability checking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

Optimizations

- Redundant edge avoidance

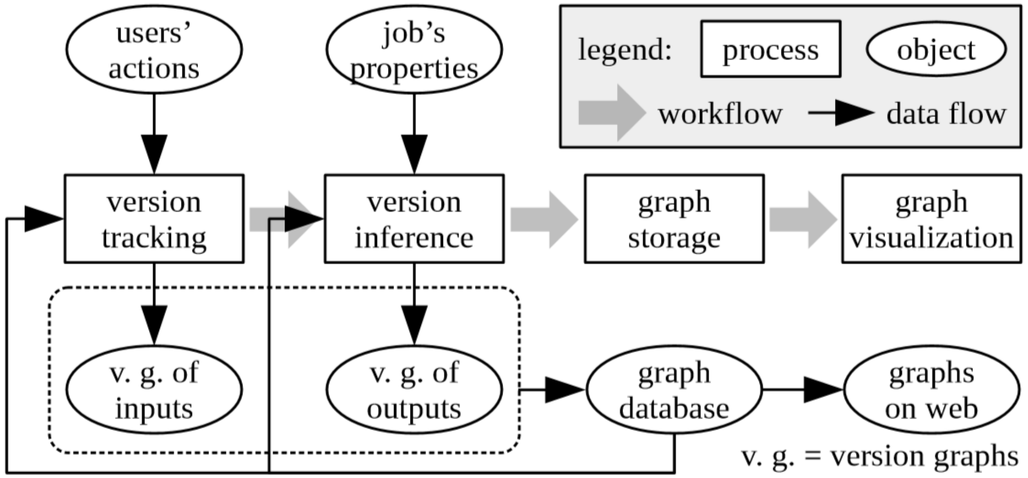


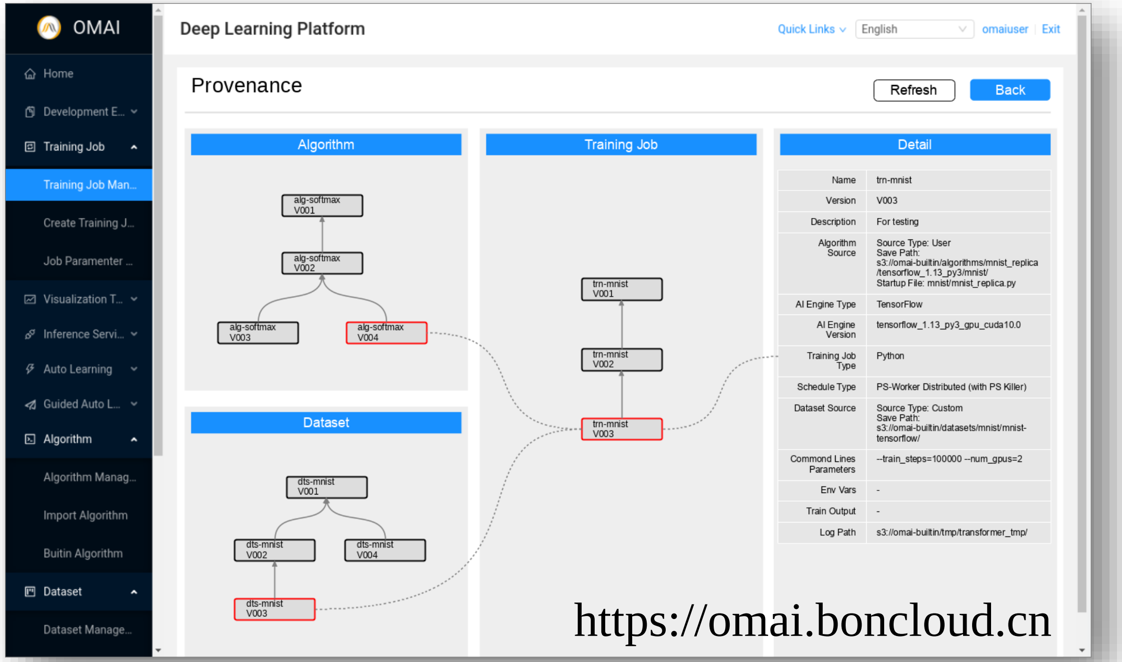
- Reverse version inference



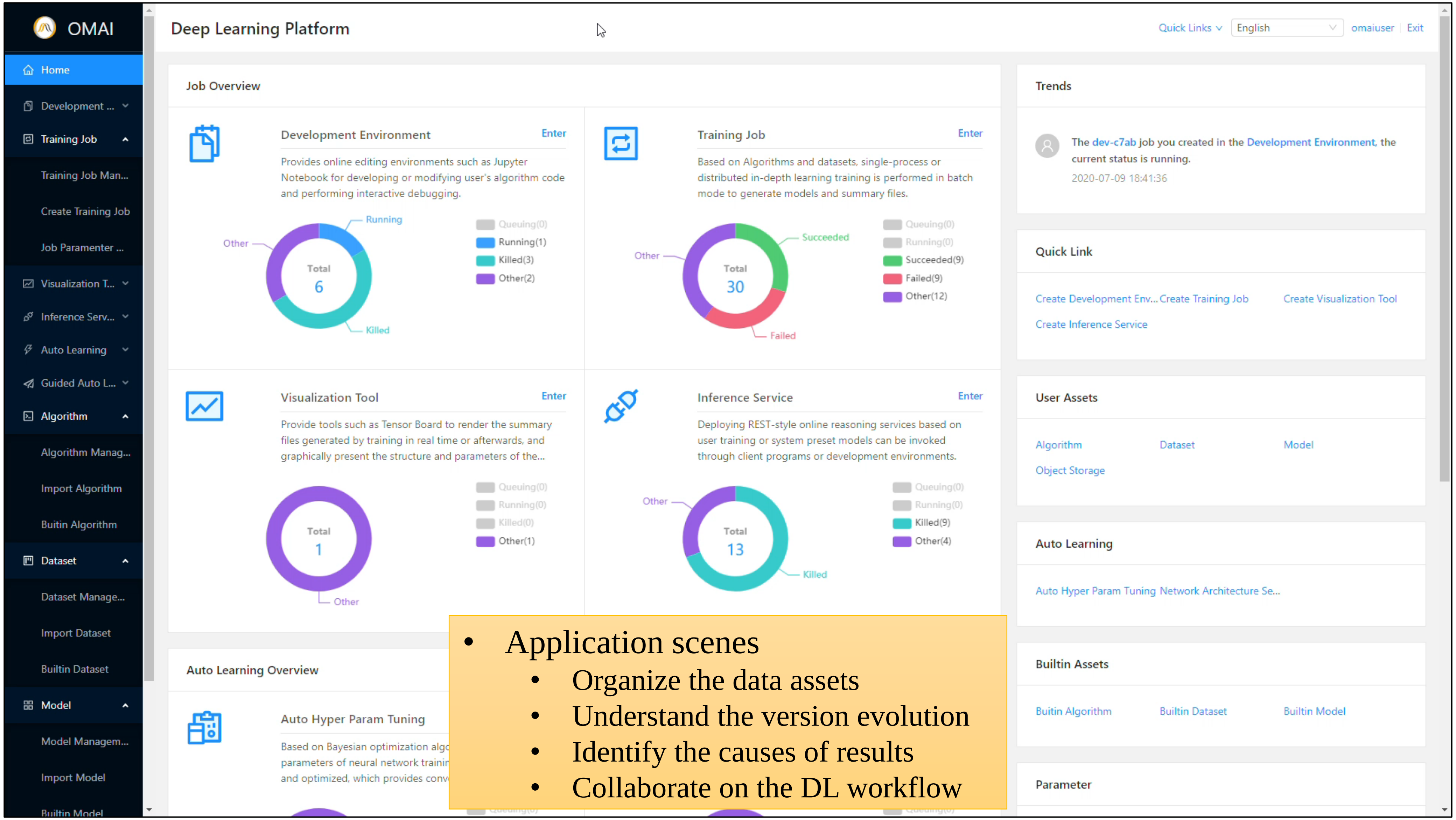
Implementation

- OMAI Deep Learning Platform
- Graph database: ArangoDB





Practice



- 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

东云睿连
Oriental Mind

Thanks!