XGraph Spec

|  |  |  |  |
| --- | --- | --- | --- |
| V1.0 | 2019.6.1 | Yushun | Spec draft |
|  |  |  |  |

[1. Introduction 3](#_Toc10206592)

[2. Function 4](#_Toc10206593)

[2.1. graph operation 4](#_Toc10206594)

[2.2. Vertex operation 4](#_Toc10206595)

[2.3. Edge operation 4](#_Toc10206596)

[2.4. Serialization 5](#_Toc10206597)

[2.5. SubGraph 5](#_Toc10206598)

[2.6. Algorithm 5](#_Toc10206599)

[3. APIs 6](#_Toc10206600)

[3.1 AddVertex 6](#_Toc10206601)

[3.2 DelVertex 6](#_Toc10206602)

[3.3 GetOP 7](#_Toc10206603)

[3.4 GetTensor 7](#_Toc10206604)

[3.5 GetInputOP 8](#_Toc10206605)

[3.6 GetConstOP 8](#_Toc10206606)

[3.7 GetOutputOP 9](#_Toc10206607)

[3.8 GetInputTensor 9](#_Toc10206608)

[3.9 GetConstTensor 10](#_Toc10206609)

[3.10 GetOutputTensor 10](#_Toc10206610)

[3.11 GetInputNum 11](#_Toc10206611)

[3.12 GetConstNum 12](#_Toc10206612)

[3.13 GetOutputNum 12](#_Toc10206613)

[3.14 AddEdge 12](#_Toc10206614)

[3.15 DelEdge 13](#_Toc10206615)

[3.16 ShowVertex 14](#_Toc10206616)

[3.17 WriteGV 14](#_Toc10206617)

[3.18 TopoSort 14](#_Toc10206618)

# Introduction

XGraph is a middle IR which are used to represent Neural Network models from different frameworks, such as Caffe, TF, PyTortch and so on. We use BGL to build XGraph while BGL is a stable and strong graph library.

Agreement:

1. Use op pointer as graph vertex
2. Use tensor pointer as graph edge
3. Only support multi-inputs and one output now(NOTE: We can support multi-outputs in the graph by adding a selector operator, its input is a tensor vector and output is one tensor)

Limitations:

1. Parallel edges are not allowed
2. Rings are not allowed.(for now)

# Function

In XGraph, we support following functions:

## 2.1. graph operation

1. Show graph vertices and edges
2. Transform graph into PNG

## 2.2. Vertex operation

1. Add vertex
2. Delete vertex
3. Get vertex
4. Get source vertex
5. Get source edge
6. Get destination vertex
7. Get destination edge
8. Get vertex in-degree
9. Get vertex out-degree
10. Show all vertices

## 2.3. Edge operation

1. Add edge
2. Delete edge
3. Get edge tensor
4. Show all edges

## 2.4. Serialization

We prepare to do serialization using protobuf:

1. Using xml to generate op and tensor proto file
2. Define graph proto to present relationship of vertex and edge
3. Not serialize extended attributes registered by compiler

## 2.5. SubGraph

TODO

## 2.6. Algorithm

1. topological sort
2. Depth-first search
3. Breadth-first search

# APIs

XGraph is a template class, we can use self-defined OP and Tensor for future, in following API declaration, we use:

1. NTypePtr to represent OP type
2. TTypePtr to represent Tensor type

## AddVertex

Add a vertex into graph

1. **Declaration**

void GetVertex(const NTypePtr& op)

1. **Parameter**

op: an op object pointer

1. **Return Value**

None

## DelVertex

Delete specified vertex from graph

1. **Declaration**

void DelVertex(const string& op\_name)

1. **Parameter**

op\_name: an op name which is unique in graph

1. **Return Value**

None

## GetOP

Get specified OP from the graph

1. **Declaration**

const NTypePtr GetOP(const string& op\_name) const

1. **Parameter**

op\_name: an op name which is unique in graph

1. **Return Value**

Return an OP pointer

## GetTensor

Get specified OP from the graph

1. **Declaration**

const TTypePtr GetTensor(const string& src\_op\_name, const string& dst\_op\_name) const

1. **Parameter**

src\_op\_name: an op name which stands the start vertex of the edge

dst\_op\_name: an op name which stands the end vertex of the edge

1. **Return Value**

Return an OP pointer

## GetInputOP

It is an overwritten func used to Get all or one input OP from specified vertex

1. **Declaration**

const vector<NTypePtr> GetInputOP(const string& op\_name) const

const NTypePtr GetInputOP(const string& op\_name, int id) const

1. **Parameter**

op\_name: vertex op name

id: input op id used to distinguish multi-inputs

1. **Return Value**

Return an OP pointer vector or an OP pointer only

## GetConstOP

It is an overwritten func used to Get all or one const OP from specified vertex

1. **Declaration**

const vector<NTypePtr> GetConstOP(const string& op\_name) const

const NTypePtr GetConstOP(const string& op\_name, int id) const

1. **Parameter**

op\_name: vertex op name

id: const op id used to distinguish multi-consts

1. **Return Value**

Return an OP pointer vector or an OP pointer only

## GetOutputOP

It is an overwritten func used to Get all or one output OP from specified vertex

1. **Declaration**

const vector<NTypePtr> GetOutputOP(const string& op\_name) const

const NTypePtr GetOutputOP(const string& op\_name, int id, int sub\_id=0) const

1. **Parameter**

op\_name: vertex op name

id: output op id used to distinguish multi-outputs

sub\_id: the number of the output with same id

1. **Return Value**

Return an OP pointer vector or an OP pointer only

## GetInputTensor

It is an overwritten func used to Get all or one input Tensor from specified vertex

1. **Declaration**

const vector<TTypePtr> GetInputTensor(const string& op\_name) const

const TTypePtr GetInputTensor(const string& op\_name, int id) const

1. **Parameter**

op\_name: vertex op name

id: input op id used to distinguish multi-inputs

1. **Return Value**

Return an Tensor pointer vector or an Tensor pointer only

## GetConstTensor

It is an overwritten func used to Get all or one const Tensor from specified vertex

1. **Declaration**

const vector<TTypePtr> GetConstTensor(const string& op\_name) const

const TTypePtr GetConstTensor(const string& op\_name, int id) const

1. **Parameter**

op\_name: vertex op name

id: const op id used to distinguish multi-consts

1. **Return Value**

Return an Tensor pointer vector or an Tensor pointer only

## GetOutputTensor

It is an overwritten func used to Get all or one output Tensor from specified vertex

1. **Declaration**

const vector<TTypePtr> GetOutputTensor(const string& op\_name) const

const TTypePtr GetOutputTensor(const string& op\_name, int id, int sub\_id=0) const

1. **Parameter**

op\_name: vertex op name

id: output op id used to distinguish multi-outputs

sub\_id: the number of the output with same id

1. **Return Value**

Return an Tensor pointer vector or an Tensor pointer only

## GetInputNum

Get input number of specified vertex

1. **Declaration**

const int GetInputNum(const string& op\_name) const

1. **Parameter**

op\_name: vertex op name

1. **Return Value**

Return input number

## GetConstNum

Get const number of specified vertex

1. **Declaration**

const int GeConstNum(const string& op\_name) const

1. **Parameter**

op\_name: vertex op name

1. **Return Value**

Return const number

## GetOutputNum

Get output number of specified vertex

1. **Declaration**

const int GeOutputNum(const string& op\_name) const

1. **Parameter**

op\_name: vertex op name

1. **Return Value**

Return output number

## AddEdge

Add an edge into the graph, it is an overwritten function.

1. **Declaration**

void AddEdge(const NTypePtr& src, const NTypePtr& dst, const tensor::TensorType& src\_ttype, const tensor::TensorType& dst\_ttype, const TTypePtr& tensor)

void AddEdge(const string& src\_op\_name, const string& dst\_op\_name, const tensor::TensorType& src\_ttype, const tensor::TensorType& dst\_ttype, const TTypePtr& tensor)

1. **Parameter**

src: souce OP pointer object of the edge

dst: destination OP pointer object of the edge

src\_op\_name: source vertex op name of the edge

dst\_op\_name: destination vertex op name of the edge

tensor: tensor ptr object

1. **Return Value**

None

## DelEdge

Delete an edge from the graph

1. **Declaration**

void DelEdge(const string& src\_op\_name, const string& dst\_op\_name)

1. **Parameter**

src\_op\_name: source vertex op name of the edge

dst\_op\_name: destination vertex op name of the edge

1. **Return Value**

None

## ShowVertex

Write all the vertex info into log file

1. **Declaration**

void ShowVertex()

1. **Parameter**

None

1. **Return Value**

None

## WriteGV

Write the graph into graphviz file and related PNG file

1. **Declaration**

void WriteGV(const string& fname\_prefix, bool dot2png)

1. **Parameter**

fname\_prefix: graphviz file prefix name

dot2png: flag to indicate whether transform graphviz file into png

1. **Return Value**

None

## TopoSort

Do topological sort for the graph

1. **Declaration**

const vector<NTypePtr> TopoSort() const

1. **Parameter**

None

1. **Return Value**

The topological sort result, it is a OP pointer vector