tf_onnx Documentation

Release

Hiroshi Sawada

Jan 24, 2018

Contents

This is an introduction tutorial to TF_ONNX. TF_ONNX is a conversion module to let a protobuffer defined on a protocol buffer another protobuffer on ONNX.

This supports not only just another straightforward conversion, but enables you to customize a given graph structure in a concise buf very flexible manner to let the conversion job very tidy.

For instance, it will support following features in a least confusing manner.

- operation conversion (controlling number of arugments)
- sub-branch pruning (removing identity node on a tensorflow protobuf)
- sub-branch evaluation (useful for quantization function)
- visualization (dot language provided by graphviz library)

Note: This is stil not a general converter from tensorflow pb to onnx pb as the combination of current onnx supported operation cannot cover the operator defined under the Nodedef object on tf-pb.

That being said, numbers of kinds of "op" on Nodedef() would be much less than seeming number of operators written in a tf-tutorial which is nearly 1000 as a function defined as python would be decomposed as a set of basic operators written in c++ on tf.

CHAPTER 1

Contents:

1.1 Installation

I am very afraid to let this library be depenedent on other big libraries such as Tensorflow or ONNX because it may give it some side effects to conversion process.

Both protocol buffer is therefore extracted from a snapshot of both.

only requirement so far is "numpy"

1.1.1 Install with docker

```
some command1
```

1.1.2 Install with virtualenv

```
some command2
```

1.2 An Explanation of relavant Protocol buffers

This page is for quick understanding of protocol buffer format.

Tensorflow Protocol Buffer

ONNX Protocol Buffer

1.3 Structure of Conversion

A conversion will be done by a tree structure and a set of higher order function on it.

1.4 Examples

This page will introduce some basic examples for conversion and a few tools to make your life easier.

As this explanation will trace example codes which are put on a `./examples/*` refer them with this.

`./example/ex1.py`

So far, if somebody needs more explanations more than what was written on a each script,

I will add more on it.

1.5 tf_onnx

1.5.1 Submodules

tf_onnx.core_

Submodules

tf_onnx.core_.node

Classes

- `Node`: Base node object.

tf_onnx.core_.node_

Submodules

tf_onnx.core_.node_.inner_

Submodules

tf_onnx.core_.node_.inner_.arg1_

Classes

- `Arithmetic`: Arithmetic class has axes and keepdims as an attribute.
- `Cast`: Cast arg1 does not have any attributes to be added on ONNX
- `Compare`: individual axis are provided in each specific class
- `F_shape`: individual axis are provided in each specific class
- `IO`: IO has no role in this graph.

Classes

- `Arg1`: this class contains just a child.
- `Arg2`: this class has two children.
- `ArgMany`: this class will be set for an operator which has more than 3 arguments.

tf_onnx.core_.node_.leaf_

Submodules

tf_onnx.core_.node_.leaf_.const

Classes

- `Const`: Const node is one of leafs which contains values on a tensor.

tf_onnx.core_.node_.leaf_.placeholder

Classes

- `Placeholder`: Placeholder is a leaf but not initialized like data on bss.

Classes

- `Leaf`: leaf node has no children on a graph.
- `Inner`: there is still no method on this abstraction level but class itself

tf_onnx.type_

Submodules

tf_onnx.type_.cast_

Functions

- `pb_to_json()`: TYPE :: pb object -> json pbjec
- `json_to_pb()`: TYPE :: json object on python -> pbobject on python -> pb_objec
- `protobuf_to_dict()`: Undocumented.

Variables

- `TYPE_CALLABLE_MAP`
- `REVERSE_TYPE_CALLABLE_MAP`

tf_onnx.type_.dict2pb

Functions

- protobuf_to_dict(): Undocumented.
- dict_to_protobuf(): Populates a protobuf model from a dictionary.

Variables

- TYPE_CALLABLE_MAP
- REVERSE_TYPE_CALLABLE_MAP

CHAPTER 2

Indices and tables

- genindex
- modindex
- search