ONNX Development Lab: Develop support for a new/updated operator

Winnie Tsang, Chin Huang, Cognitive OpenTech

Data and AI
Open Source Dojo

Outline

- Unit 1: Setup and verify development dependencies
- Unit 2: Develop support for a new/updated operator

Unit 1: Setup development dependencies

- **Goal of the unit: At the end of unit 1, you will learn how to install and verify the projects that are required for the ONNX Tensorflow converter development**
- Development dependencies
    - System packages
    - ONNX master
    - Tensorflow 2.1
    - ONNX-TF master
- Step 1.1: Please make sure you have finished the setup and verified the dependencies in the earlier "Development Environment" session. If not, please go to https://github.com/chinhuang007/onnx-dojo/blob/master/docs/day1_afternoon/onnx_dev_env_dojo_2020.pdf and complete the setup.

Unit 2: Develop support for a new/updated operator

- **Goal of the unit: At the end of unit 2, you will learn all the steps that required to support a new/updated ONNX operator in ONNX-Tensorflow converter. In this unit, we are going to use Constant operator in Opset 12 as our target operator to support.**
- Step 2.1: Study the specification of Constant in ONNX to identify changes in opset 12 https://github.com/onnx/onnx/blob/master/docs/Operators.md#Constant
    - Compare Constant-12 with the previous version Constant-11 https://github.com/onnx/onnx/blob/master/docs/Changelog.md#Constant-11
    - Constant-12 accept python primitive type like float, int and string as input attributes in addition to tensor and sparse_tensor in Constant-11

Unit 2: Develop support for a new/updated operator

- Step 2.2: Identify the potential Tensorflow operator to support the new changes in Constant-12 https://www.tensorflow.org/api_docs/python/tf/constant
    - Argument “value” in tf.constant accept constant value or list of any Tensorflow DataType
        - Therefore tf.constant is still efficient to perform the same capability of ONNX Constant-12 op
- Step 2.3: Create your PR development workspace
    - Verify your ONNX package location is point to your ONNX directory not to an egg file
        - *pip list | grep onnx*
        - If the location is point to an egg file, then please reinstall it with the following commands
        - *cd <your-onnx-directory> ; pip install -e .*

Unit 2: Develop support for a new/updated operator

- If you didn't create a fork of ONNX-Tensorflow yet, please create one by clicking Fork 156 button on the top right corner of ONNX-Tensorflow repository
- Clone your fork
  - *git clone https://github.com/<your-git-user-name>/onnx-tensorflow.git*
  - Verify: *cd onnx-tensorflow ; ls*
- Setup the upstream remote
  - *git remote add upstream https://github.com/onnx/onnx-tensorflow.git*
  - Verify: *git remote -v*
- Verify your fork master is up-to-date by navigate your browser to your fork https://github.com/<your-git-user-name>/onnx-tensorflow
  - If your master is up-to-date then you should find "This branch is even with onnx:master" display on the top.

Unit 2: Develop support for a new/updated operator

- If you don't see the above message then your master is out-of-date, please run the following commands to update it
  - *git fetch upstream*
  - *git merge upstream/master*
  - *git push origin master*
- Create a branch under your fork as your development workspace
  - *git checkout -b <your-branch-name>*
  - Verify: *git branch*
- Use your branch to build onnx-tf
  - *pip install -e .*
  - Verify: *pip list | grep onnx-tf*

Unit 2: Develop support for a new/updated operator

- Step 2.4: Update test_constant unit test in onnx-tensorflow/test/backend/test_node.py
    - Let's only focus on adding the int64 attribute support for Constant here
    - Add a test case in test_constant to check does the existing Constant handler accept a constant int64 value as an attribute
    - When you are ready to test run your updated test_constant, please run the following command:
        - *python test_node.py TestNode.test_constant*

Unit 2: Develop support for a new/updated operator

- Step 2.5: Update Constant handler in onnx-tensorflow/onnx_tf/handlers/backend/constant.py
    - Define version_12 method to handle Constant in opset 12 and above
        - Need to handle everything opset 11 support
        - Plus the newly added attributes like “value_int”
    - Register your updated handler to onnx-tensorlfow/onnx_tf/opset_version.py file
        - Run onnx-tensorlfow/onnx_tf/gen_opset.py
            - *python gen_opset.py*
            - Verify: *cat opset_version.py | grep Constant*
    - Test/debug your updated handler by running your updated Constant unit test in onnx-tensorflow/test/backend/TestNode.py on step 2.4
        - *python test_node.py TestNode.test_constant*

Unit 2: Develop support for a new/updated operator

- After successfully run the unit test of Constant in test_node.py then please verify your change doesn't create any new error in other tests under onnx-tensorlfow/test/backend folder.
  - *python test_node.py*
  - *python test_dynamic_shape.py*
  - *python test_model.py*
  - *python test_onnx_backend.py*
  - *python ../test_cli.py*

  Note: test_onnx_backend.py typically takes between 20 to 40 minutes to complete, depending on hardware configurations

- Step 2.6: Update support status report for Constant
  - Run onnx-tensorflow/onnx_tf/gen_status.py to update the support status
    - *python gen_status.py –v master*
  - Verify: *cat ../doc/support_status.md | grep Constant*

Unit 2: Develop support for a new/updated operator

- Step 2.7: Verify all changed files follow the recommended code format
    - Follow instructions for code standard format on https://github.com/onnx/onnx-tensorflow#code-standard
- Step 2.8: Commit all the changes to your branch in your fork
    - *git status*
    - *git add **
    - *git commit*
    - *git push origin <your-branch-name>*

Unit 2: Develop support for a new/updated operator

- Step 2.9: Create Pull Request(PR) in ONNX-Tensorflow Repository
    - Navigate your browser to https://github.com/onnx/onnx-tensorflow,
    - Click on the “Compare and pull request” button
    - Click on the “Create pull request“ button
    - Type a title and description of your pull request
    - Click on the “Create pull request“ button to submit it

    Note: If you can’t find the “Compare and pull request” button then run the following steps:

    - Click on the “Pull requests” tab on the top of the page
    - Click on the “New pull request” button
    - Click on the “compare across forks” link
    - Leave the base repo as the master branch and change the head repo to your fork and your branch
    - Click on the “Create pull request“ button

Unit 2: Develop support for a new/updated operator

- Step 2.10: Wait for review and address comment
    - If changes is required in your PR, modify code in your branch then run the following git commands:
        - *git status*
        - *git add **
        - *git commit --amend*
        - *git push -f origin <your-branch-name>*

Unit 2: Develop support for a new/updated operator

- If rebase is required, then run the following git commands
  - *git stash* # if there is uncommitted changes on the current branch then stash it else skip to next step
  - *git fetch upstream*
  - *git checkout master*
  - *git merge upstream/master*
  - *git push origin master*
  - *git checkout <your-branch-name>*
  - *git rebase origin/master*
  - *git stash pop* # if there is a stash then pop it now else skip to next step
  - *git commit --amend* # if there is no change to the commit then skip to next step
  - *git show -1*
  - *git push -f origin <your-branch-name>*