ONNX Pre-processing WG

Update - Oct 21, 2021
Joaquin Anton (NVIDIA)
The problem

Lack of standardized preprocessing primitives, portability issues

Hard to deploy pre-trained models to optimized runtimes.
Preprocessing Often implemented in Python, with libraries such as Pillow, Numpy, OpenCV
Roadmap

- Make data preprocessing part of ONNX
  - Data preprocessing to be distributed with the ONNX model
  - Easy to deploy
- Standardize definition of pre-processing primitives
  - Portable across implementations
  - Focus on vision networks first
  - Extend to other data domains later (e.g. audio)
  - Extend operator support to cover most popular networks
Proof of concept - ResNet 50

Preprocessing is typically defined in Python

```python
from PIL import Image

def preprocess(image):
    # resize so that the shorter side is 256, maintaining aspect ratio
    def resize(image, min_len):
        image = Image.fromarray(image)
        ratio = float(min_len) / min(image.size[0], image.size[1])
        if image.size[0] > image.size[1]:
            new_size = (int(round(ratio * image.size[0])), min_len)
        else:
            new_size = (min_len, int(round(ratio * image.size[1])))
        image = image.resize(new_size, Image.BILINEAR)
        return np.array(image)

    # Image resizing
    width, height = image.size
    new_width = int(np.ceil(float(width) / 2) * 2)
    new_height = int(np.ceil(float(height) / 2) * 2)
    image = image.resize((new_width, new_height), Image.BILINEAR)

    # Crop centered window 224x224
    h, w, c = image.shape
    x = int((w - new_width) / 2)
    y = int((h - new_height) / 2)
    image = image[y:y + new_height, x:x + new_width]

    # Transpose
    image = np.array(image).transpose(2, 0, 1)

    # Normalize
    mean_vec = np.array([0.485, 0.456, 0.406])
    stddev_vec = np.array([0.229, 0.224, 0.225])
    for i in range(image.shape[0]):
        norm_img_data[i] = (image_data[i] - mean_vec[i]) / stddev_vec[i]
    norm_img_data = norm_img_data.astype('float32')

    # Add channel
    norm_img_data = norm_img_data.reshape(1, 3, 224, 224).astype('float32')
    return norm_img_data
```

```python
session = onnxruntime.InferenceSession('rn50-preprocessing.onnx', None)
input_data = preprocess(image_data)
raw_result = session.run([], {"name": input_data})
```

Input image

Small artifacts due to different implementations

```python
np.abs(data_onnx - data_numpy) * 255
```
Data pre-processing within ONNX

- Some ideas being discussed
  - Pre-processing pipeline to be stored as a standalone model
  - The actual model and the preprocessing model are combined together
  - For more than one sample, a special control flow operator could be used to apply the preprocessing subgraph to individual samples and combine them to a uniform batch

Just an idea for now:

```python
session = onnxruntime.InferenceSession('rn50v2_model.onnx', None,
preprocessing='rn50v2_preprocessing.onnx')
raw_result = session.run([], {'data': [sample1, sample2, ...]})
```
Initial Operator support

- Vision networks:
  - Classification: ResNet (and ResNext)
  - Detection & Segmentation: SSD, Mask R-CNN

- Operators:
  - Resize: Extend
    - Interpolation type \{nearest-neighbor, bilinear, bicubic, \texttt{triangular}, lanczos\}
    - Resize policy \{stretch, not-larger, not-smaller\}
  - Color space conversion: New operator
    - RGB to BGR, etc
  - Slice: OK
  - Cast: OK
  - Normalize (Sub/Div): OK
  - Transpose: OK
  - Pad: OK
  - Shape: OK
Get involved!

- Slack channel: [https://slack.lfai.foundation](https://slack.lfai.foundation) and join **onnx-preprocessing**
- Monthly WG meetings (see slack channel for announcements)