ONNX Pre-processing WG

Update - Oct 21, 2021

Joaquin Anton (NVIDIA)

The problem

Lack of standardized preprocessing primitives, portability issues

Bilinear filter – OpenCV vs Pillow

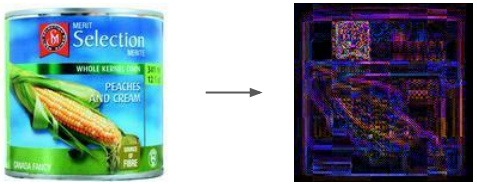
Pillow's "bilinear" interpolation is actually triangular

Bicubic filter – TensorFlow 1 vs Pillow

Different definition of pixel centers when resampling

https://hackernoon.com/how-tensorflows-tf-image-resize-stole-60-days-of-my-life-aba5eb093f35

JPEG decoding - libjpeg-turbo:
"Fancy upsampling" ON (default) vs OFF



abs_diff * 10

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

Roadmap

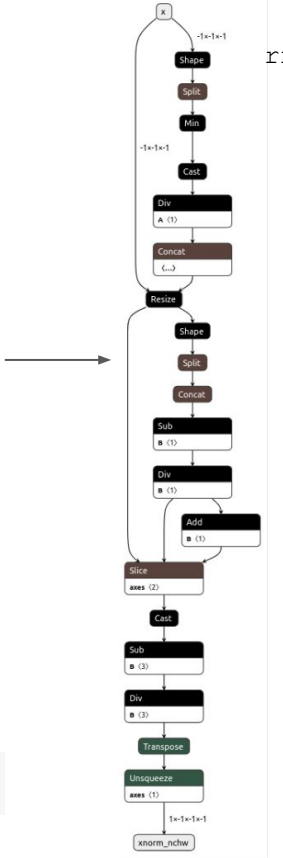
- Make data preprocessing part of ONNX
  - Data preprocessing to be distributed with the ONNX model
  - Easy to deploy
- Standardize definition of pre-processing primitives
  - Portable across implementations
  - Focus on vision networks first
  - Extend to other data domains later (e.g. audio)
  - Extend 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 image_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 = image_resize(image, 256)

    # Crop centered window 224x224
    def crop_center(image, crop_w, crop_h):
        h, w, c = image.shape
        start_x = w//2 - crop_w//2
        start_y = h//2 - crop_h//2
        return image[start_y:start_y+crop_h, start_x:start_x+crop_w, :]
    image = crop_center(image, 224, 224)

    # transpose
    image = image.transpose(2, 0, 1)

    # convert the input data into the float32 input
    img_data = image.astype('float32')

    # normalize
    mean_vec = np.array([0.485, 0.456, 0.406])
    stddev_vec = np.array([0.229, 0.224, 0.225])
    norm_img_data = np.zeros(img_data.shape).astype('float32')
    for i in range(img_data.shape[0]):
        norm_img_data[i,:,:] = (img_data[i,:,:]/255 - mean_vec[i]) / stddev_vec[i]

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

```python
session = onnxruntime.InferenceSession('resnet50v2/resnet50v2.onnx', None)
input_data = preprocess(image_data)
raw_result = session.run([], {input_name: input_data})
```



rn50-preprocessing.onnx

```python
preprocessing = onnxruntime.InferenceSession('rn50-preprocessing.onnx', None)
session = onnxruntime.InferenceSession('resnet50v2/resnet50v2.onnx', None)
input_data = preprocessing_session.run([], {'x': np.array(image_data)})[0]
raw_result = session.run([], {input_name: input_data})
```

Input image

`np.abs(data_onnx - data_numpy) * 255`

Small artifacts due to different implementations

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:

```
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, **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 and join **onnx-preprocessing**
- Monthly WG meetings (see slack channel for announcements)