ONNX pre/post processing and featurization WG

Kick-off meeting - Aug 11, 2021

Agenda

- Introduction
  - Member introductions
  - Meeting date/time and frequency
  - Preferred method of communication
- Define problem statement
  - Motivation
  - Gathering suggestions
  - Arrive at a consensus
- Action items for following meeting

Problem statement

- Lack of standardized definition for data pre-processing primitives
- Often implemented in Python, using different libraries (OpenCV, Pillow, TF data, DALI)
- Portability issues: subtle differences in a preprocessing primitive definition can result in degradation of the network performance. Effectively, the model is coupled with the preprocessing definition used at training, which might not be available at inference time
- Hard to deploy pre-trained models to optimized runtimes

Problem statement

Bilinear filter – OpenCV vs Pillow

Pillow's "bilinear" interpolation is actually triangular

Bicubic filter – TensorFlow vs Pillow

Different definition of pixel centers when resampling

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

Problem statement

STFT (Short-term Fourier Transform) - Torchaudio vs DALI
(center=False, nfft > win_len)

Torchaudio (and librosa):

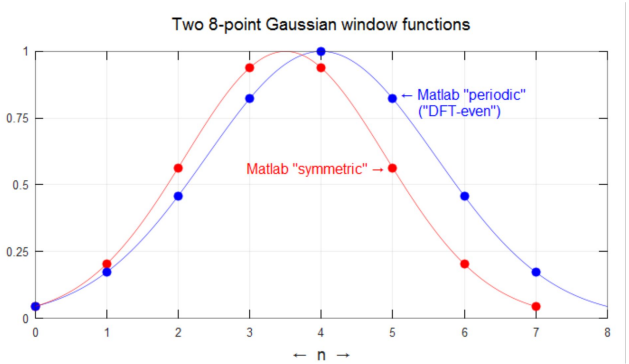
- Pads window_fn from win_len to nfft
- Extracts windows of nfft size applying the padded window_fn
- Calculates FFT

DALI:

- Extracts windows of win_len, applying the original window_fn
- Centers and pads on each side the extracted window to nfft size
- Calculates FFT

When not centering the windows at multiples of the window step, the effective position of the extracted windows will be different

Window functions - Torchaudio vs Kaldi



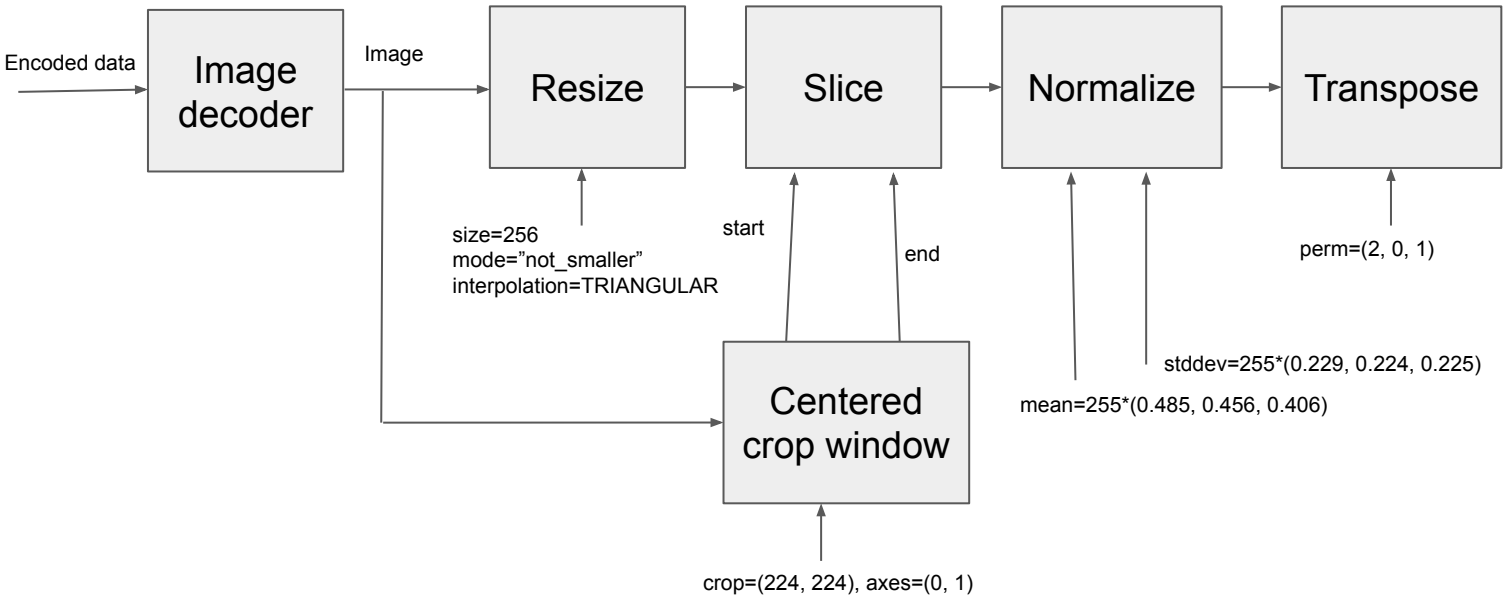
Torchaudio
Generates periodic window functions by default

Kaldi (and DALI):
Generates symmetric window functions by default

Goal

- Data preprocessing to be part of the ONNX graph
- Standardized definition of data preprocessing primitives for both inference and training
- ONNX operators to support data pipelines for most popular networks (TBD)
  - Image Classification: Resnet, Resnext, …
  - Detection/Segmentation: SSD, MaskRCNN, …
  - Speech Recognition: Jasper, RNN-T, …
  - …
- Collectively come up with a list of networks and operators we want to support.

Practical Example: ResNet (inference)



Practical Example: ResNet (training)

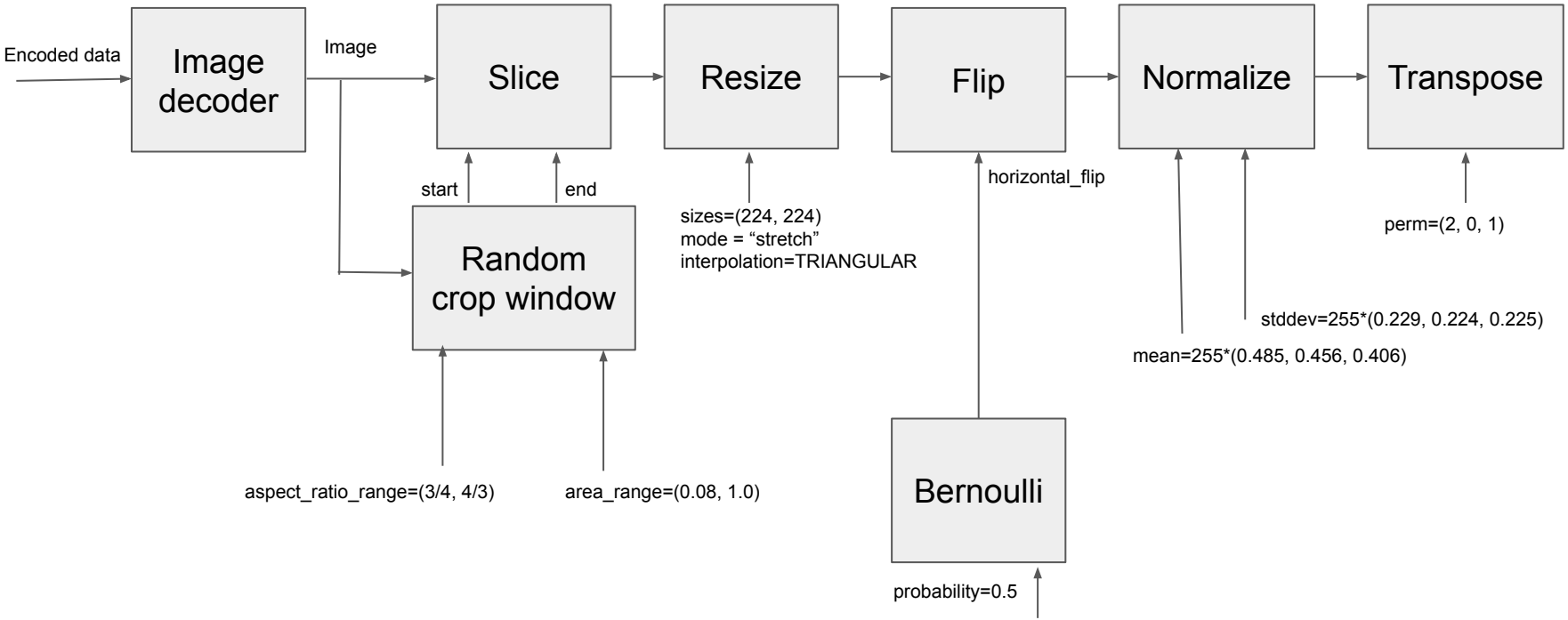


Image decoder

- Should image decoding be part of the ONNX graph?
- Should support at least most common formats: JPEG, PNG, TIFF, BMP, PPM
- Hard, as JPEG decoding is not well standardized. On the other hand, there can be noticeable differences between decoding libraries.

Centered crop window (inference)

- Produce slice start and end coordinates given the input shape and the desired crop dimensions
- Trivial to implement with arithmetic operators (**ai.onnx.Sub**, **ai.onnx.Add**, **ai.onnx.Div**)

Random crop window (training)

- New operator
- Produces slice start and end coordinates given the shape of the input and desired ranges for aspect ratio and relative area for the random cropping window.
- Takes the shape of an image as an input (or the image itself)
- Optionally accepts a random seed

Slice

- Already available as **ai.onnx.Slice**

Resize

- Already available as **`ai.onnx.Resize`**
- Coordinate transformation mode argument should be chosen so that Resize and flip are commutative (`Flip(Resize(img)) == Resize(Flip(img))`). Current default (half pixel) is OK.
- Currently supports nearest neighbor, linear, and cubic interpolation. Consider adding support for triangular, gaussian, lanczos3.
- Consider allowing different downscaling and upscaling interpolation methods
- Need a way to control the resizing policy/mode:
  - Stretch
  - Not-larger (largest dimension is resized to the given size, keeping aspect ratio)
  - Not-smaller (smallest dimension is resized to the given size, keeping aspect ratio)
- Resize policy could be either incorporated into Resize, or be a separate operator that works with shapes and produces the final shape to be consumed by Resize.

Bernoulli (training)

- Already available as **`ai.onnx.Bernoulli`**

Flip (training)

- Can be achieved with **`ai.onnx.Slice`** and a negative step for the flipped dimension

```
starts = ai.onnx.Where(should_flip, INT_MAX, INT_MIN)

ends =  ai.onnx.Where(should_flip, INT_MIN, INT_MAX)

steps = ai.onnx.Where(should_flip, -1, 1)
```

`axes = [1]` - horizontal flip, assuming HWC layout

Normalize

- Can be achieved by combining **ai.onnx.Sub** and **ai.onnx.Div** (or **ai.onnx.Mul**)

X_norm = ai.onnx.Div(ai.onnx.Sub(x, mean), stddev)

or

X_norm = ai.onnx.Mul(ai.onnx.Sub(x, mean), inv_stddev)

Transpose

- Already available as **ai.onnx.Transpose**
- May be needed or not depending on what layout the network expects

Open questions

- Data domains:
  - Image
  - Coordinates (bounding boxes, mask polygons, etc)
  - Video
  - Audio
  - …
- Notion of batch of samples? Operators that work on a batch level. Examples:
  - Intra-batch normalization
  - Mosaic augmentation
- Notion of data layout? For example, in DALI tensors carry layout information as metadata (e.g. “HWC”). Operators can choose implementation based on the layout or ignore certain dimensions (channels) depending on the nature of the operator. Examples:
  - Resize might ignore the channels dimension.
  - Mel filter bank applied to spectrogram: Where is the “frequency” dimension? Time-major vs. frequency-major
- ...

Actions for next meeting

- Each organization represented in the group comes up with a list networks that they would like to be supported, sorted by priority.
- ...