ONNX Pre-processing WG

Update - June 24, 2022

Joaquin Anton (NVIDIA)

The problem

Preprocessing steps

float32
[1, 3, 224, 224]

ONNX Model

e.g.: Crop, Resize, Normalize, Transpose

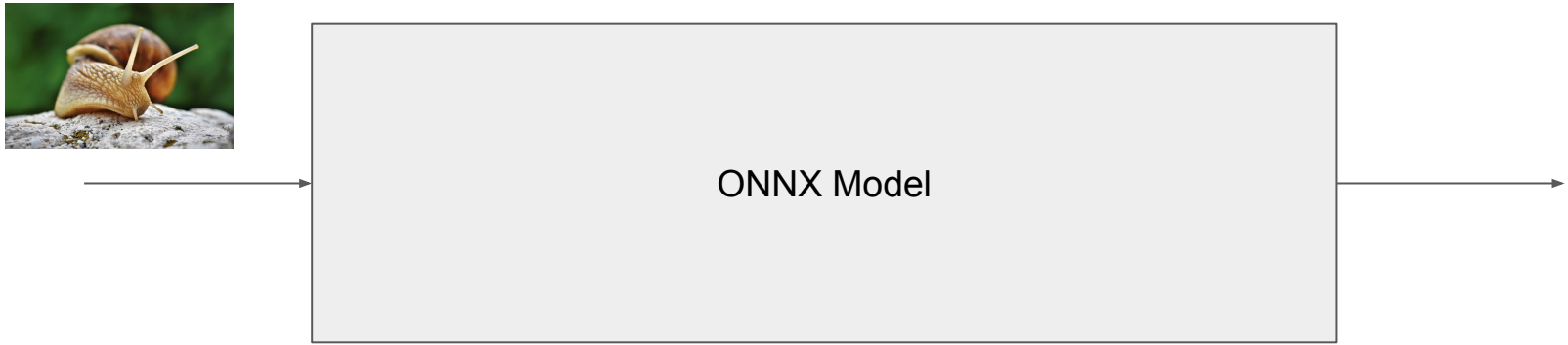
e.g.: ResNet-50

The problem

PIL.Image.resize(..., Image.BILINEAR)

cv2.resize(..., interpolation=cv2.INTER_LINEAR)

Goal



- Make data preprocessing part of ONNX
- Standardize definition of pre-processing primitives

Roadmap

- [COMPLETED Dec 2021] Composition
- [COMPLETED Apr 2022] Batch processing
- [IN PROGRESS] Identification and tagging of preprocessing subgraphs
- [IN PROGRESS] Publish a first end-to-end example: ResNet-50
  - Resize extensions: antialias filter, aspect ratio policy
  - CenterCropPad: Higher level API implemented in terms of existing ops
  - Publish to ONNX model zoo

Composition: ONNX compose utils

```python
import onnx

model1 = onnx.load('path/to/model1.onnx')
# agraph (float[N] A, float[N] B) => (float[N] C, float[N] D)
#   {
#      C = Add(A, B)
#      D = Sub(A, B)
#   }

model2 = onnx.load('path/to/model2.onnx')
#   agraph (float[N] X, float[N] Y) => (float[N] Z)
#   {
#      Z = Mul(X, Y)
#   }

combined_model = onnx.compose.merge_models(
    model1, model2,
    io_map=[('C', 'X'), ('D', 'Y')]
)
```

Batch processing: SequenceMap function

sequence<uint8[H, W, 3]>

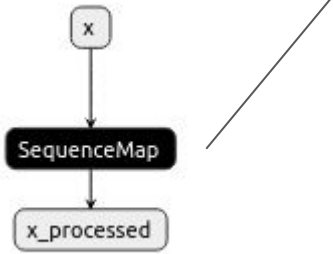
preprocessing

sequence<uint8[224, 224, 3]>

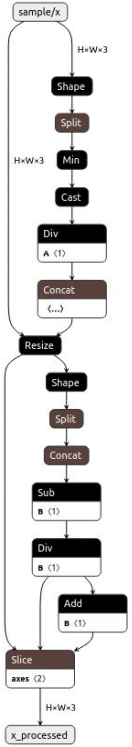
concatenate

uint8[N, 224, 224, 3]

Batch processing: SequenceMap function



body



INPUTS

x	name: **x**
	type: **sequence<uint8[H,W,3]>**

OUTPUTS

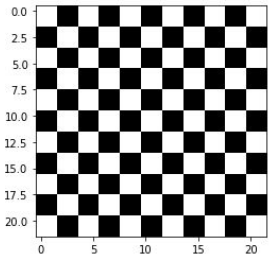
x_processed	name: **x_processed**
	type: **sequence<uint8[H,W,3]>**

Identification: tagging preprocessing subgraphs

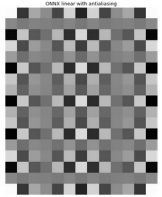
```
m = onnx.parser.parse_model('''
<
  ir_version: 8,
  opset_import: ["": 16, "local": 1],
  ...
  metadata_props = [ "preprocessing_fn" : "local.preprocess" ]
>
agraph (uint8[H, W, C] x => uint8[H, W, C] y)
{
    x_processed = local.preprocess(x)
    y = Identity(x_processed)
}

<
  opset_import: [ "" : 16 ],
  domain: "local",
>
preprocess(x) => (x_processed)
{
    x_processed = Identity(x)\
}
''')
```

Resize: Antialias and keep aspect ratio policy

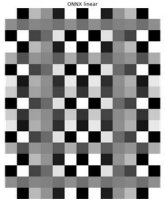
Optional antialias filter when downscaling



Resize(..., antialias=1)



Resize(..., antialias=0)



keep_aspect_ratio_policy - Treats target size as maximum or minimum,

- Values:
  - "stretch" (default) - input: (100, 50) target: (200, 200) -> output: (200, 200)
  - "not_larger" - input: (100, 50) target: (200, 200) -> output: (200, 100)
  - "not_smaller" - input: (100, 50) target: (200, 200) -> output: (400, 200)

CenterCropPad

- Implemented in terms of Slice and Pad
- Typically used in image data pipelines

Get involved!

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