ONNX & Edge Deployments
Open Neural Network Exchange

Thom Lane
28th August 2018
Agenda

1. What is ONNX?
2. Creating/finding ONNX models
3. Visualizing ONNX models
4. Deploying ONNX models
5. Optimizing ONNX models
What is ONNX?
What is ONNX?

*Open Neural Network Exchange Format*
ONNX Partners
Creating/finding ONNX models
Where can you get ONNX models from?

- Model Zoo
- Train Your Own
- Fine-tuning

Fine Tuning by Emma Mitchell, Teacher by Gregor Cresnar, take by Adrien Coquet from the Noun Project
Where can you get ONNX models from?

Model Zoo
Train Your Own
Fine-tuning

Apache MXNet on AWS SageMaker

Fine Tuning by Emma Mitchell, Teacher by Gregor Cresnar, take by Adrien Coquet from the Noun Project
## Which model to choose for edge?

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Computation</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy on ImageNet (top-1)</td>
<td>Millions of Mult-Adds</td>
<td>Millions of Parameters</td>
</tr>
<tr>
<td>VGG16</td>
<td>71.5%</td>
<td>15300</td>
<td>138</td>
</tr>
<tr>
<td>1.0 MobileNet-224</td>
<td>70.6%</td>
<td>569</td>
<td>4.2</td>
</tr>
</tbody>
</table>
Depthwise Separable Convolution

Regular Convolution

3x10x10

16x10x10

3x3(x3) Convolution

# of params: 432
# of computations: 43,200

Depthwise Separable Convolution

3x10x10

3x10x10

16x10x10

3x3(x1) Convolution

1x1(x3) Convolution

# of params: 27+48 = 75
# of computations: 2,700+4,800 = 7,500
MobileNet Example

MobileNet from ONNX Model Zoo (pretrained on ImageNet)

Fine-tune on CALtech101

Apache MXNet on AWS SageMaker

Fine Tuning by Emma Mitchell, Teacher by Gregor Cresnar, take by Adrien Coquet from the Noun Project
Apache MXNet Overview

- Scalable
- Debuggable
- Flexible
- Optimized libraries
- 7 frontend languages
- Portable

Speech Bubble by Weltenraser, Scale by Ben Davis, Bug by Nociconist, Mobile by Rafael Garcia Motta, flexible by AdbA Icons from the Noun Project
```python
1 from multiprocessing import cpu_count
2 import mxnet as mx
3 from mxnet.gluon.data.vision.transforms import Compose, RandomResizedCrop, ToTensor
4
5 # define dataset and transforms
6 train_dataset = mx.gluon.data.vision.CIFAR10(train=True)
7 transform_fn = Compose([RandomResizedCrop(32), ToTensor()])
8 train_dataset_w_aug = train_dataset.transform_first(transform_fn)
9 train_dataloader = mx.gluon.data.DataLoader(dataset=train_dataset_w_aug, batch_size=128, shuffle=True, num_workers=cpu_count(), last_batch="keep")

15 # define model and initialize its parameters
16 context = mx.gpu() # or mx.cpu()
17 net = mx.gluon.model_zoo.vision.resnet18_v2(pretrained=False)
18 net.initialize(mx.initializer.Xavier(), context)

20 # define loss to optimize
21 loss_fn = mx.gluon.loss.SoftmaxCrossEntropyLoss()

23 # define optimizer and trainer
24 optimizer = mx.optimizer.SGD(learning_rate=0.01, momentum=0.9)
25 trainer = mx.gluon.Trainer(params=net.collect_params(), optimizer=optimizer)
26
27 # train for multiple epochs
```
# define model and initialize its parameters
context = mx.gpu() # or mx.cpu()
net = mx.gluon.model_zoo.vision.resnet18_v2(pretrained=False)
net.initialize(mx.initializer.Xavier(), context)

# define loss to optimize
loss_fn = mx.gluon.loss.SoftmaxCrossEntropyLoss()

# define optimizer and trainer
optimizer = mx.optimizer.SGD(learning_rate=0.01, momentum=0.9)
trainer = mx.gluon.Trainer(params=net.collect_params(), optimizer=optimizer)

# train for multiple epochs
for epoch_idx in range(10):
    for data, label in train_dataloader:
        # transfer batch from cpu memory to gpu memory
data = data.as_in_context(context)
label = label.as_in_context(context)
        # forward pass (constructing computational graph)
with mx.autograd.record():
prediction = net(data)
        loss = loss_fn(prediction, label)
        # backward pass (backpropagating gradients through graph)
loss.backward()
        # update parameters using gradients (and scale appropriately)
trainer.step(batch_size)
        # print loss (and block async computations)
print("Loss: {}".format(loss.sum().asscalar()))
MobileNet Example

MobileNet from ONNX Model Zoo (pretrained on ImageNet)

Fine-tune on CALtech101

Apache MXNet on AWS SageMaker

Fine Tuning by Emma Mitchell, Teacher by Gregor Cresnar, take by Adrien Coquet from the Noun Project
Visualizing ONNX models
How to visualize ONNX models?

[Image of a diagram showing a network with nodes labeled BatchNormalization, Pad, and Conv.

NETRON
lutzroeder.github.io/netron]
Deploying ONNX models
How to deploy ONNX models?

- AWS SageMaker
- AWS Fargate with Model Servers
- AWS GreenGrass
- Custom Deployments

Raspberry Pi by Ben Davis, clouds by Viktor Vorobyev from the Noun Project
The Greengrass Core is the runtime that enables the local execution of AWS Lambda, messaging, device shadows, and security. The Greengrass Core interacts directly with the cloud.

Any device that uses the IoT Device SDK can be configured to interact with Greengrass Core via the local network.
AWS GreenGrass Group Deployments

- Device
- Device Resource
- ML Model Resource
- Lambda function
- Subscription
Optimizing ONNX models
How to optimize ONNX models?

1. Use half-precision (float16) if possible: e.g. Mali-GPU
2. Use quantization with calibration if possible (*experimental*)
3. Compile model with TVM Stack
What type of optimizations?

NNVM: *Graph* Optimizations
- Pruning
- Fusing

TVM: *Tensor* Optimizations
- Tiling
- Vectorization
What type of optimizations?

NNVM: *Graph* Optimizations

- Pruning
  - Conv
  - Dropout
  - Conv

- Fusing
  - Conv
  - Conv

TVM: *Tensor* Optimizations

- Tiling
- Vectorization
What type of optimizations?

**NNVM: Graph Optimizations**
- Pruning
- Fusing
  - Conv
  - Relu
  - Conv
  - Conv with Relu

**TVM: Tensor Optimizations**
- Tiling
- Vectorization
What type of optimizations?

NNVM: *Graph* Optimizations
- Pruning
- Fusing

TVM: *Tensor* Optimizations
- Tiling
  - $N \times C \times H \times W$ → $N \times (C/16) \times H \times W \times 16$
- Vectorization
What type of optimizations?

**NNVM: Graph Optimizations**
- Pruning
- Fusing

**TVM: Tensor Optimizations**
- Tiling
- Vectorization
  - $1 + 3 = 4$
  - $2 + 2 = 4$
  - $1 + 0 = 1$
  - $1 + 1 = 2$
  - $1 + 3 = 4$
  - $2 + 2 = 4$
  - $1 + 0 = 1$
  - $1 + 1 = 2$
Summary

1. Creating/finding ONNX models
   • ONNX Model Zoo
   • And fine-tune with Apache MXNet and AWS SageMaker

2. Visualizing ONNX models
   • Netron

3. Deploying ONNX models
   • AWS GreenGrass

4. Optimizing ONNX models
   • TVM Stack
Thanks!

And don’t forget to check out:
https://medium.com/apache-mxnet