ONNX & ONNX Runtime

Weixing Zhang

Microsoft AI Frameworks
Agenda

- What is ONNX
- ONNX @ Microsoft
- What is ONNX Runtime
- How to create ONNX models
Open and Interoperable AI
Open Neural Network Exchange

Open format for ML models

github.com/onnx
Key Design Principles

• Support DNN but also allow for traditional ML

• Flexible enough to keep up with rapid advances

• Compact and cross-platform representation for serialization

• Standardized list of well defined operators informed by real world usage
ONNX Spec

- File format
- Operators
File format

Model
• Version info
• Metadata
• Acyclic computation dataflow graph

Graph
• Inputs and outputs
• List of computation nodes
• Graph name

Computation Node
• Zero or more inputs of defined types
• One or more outputs of defined types
• Operator
• Operator parameters
Data types

- **Tensor type**
  - Element types supported:
    - int8, int16, int32, int64
    - uint8, uint16, uint32, uint64
    - float16, float, double
    - bool
    - string
    - complex64, complex128

- **Non-tensor types:**
  - Sequence
  - Map

```protobuf
text = "message TypeProto {
  message Tensor {
    optional TensorProto.DataType elem_type = 1;
    optional TensorShapeProto shape = 2;
  }
  // repeated T
  message Sequence {
    optional TypeProto elem_type = 1;
  }
  // map<K,V>
  message Map {
    optional TensorProto.DataType key_type = 1;
    optional TypeProto value_type = 2;
  }
}

oneof value {
  Tensor tensor_type = 1;
  Sequence sequence_type = 4;
  Map map_type = 5;
}
""
```
Operators

An operator is identified by <name, domain, version>

Core ops (ONNX and ONNX-ML)
- Should be supported by ONNX-compatible products
- Generally cannot be meaningfully further decomposed
- Currently 124 ops in ai.onnx domain and 18 in ai.onnx.ml
- Supports many scenarios/problem areas including image classification, recommendation, natural language processing, etc.

Custom ops
- Ops specific to framework or runtime
- Indicated by a custom domain name
- Primarily meant to be a safety-valve
Functions

- Compound ops built with existing primitive ops
- Runtimes/frameworks/tools can either have an optimized implementation or fallback to using the primitive ops
Agenda

✓ What is ONNX

☐ ONNX @ Microsoft

☐ What is ONNX Runtime

☐ How to create ONNX models
LOTS of internal teams and external customers

LOTS of models from LOTS of different frameworks

Different teams/customers deploy to different targets
ONNX @ Microsoft

**Platforms**
- AzureML
- WinML
- ML.Net

**Products**
- Bing
- Microsoft Cognitive Services
- Office 365
- LinkedIn ads
- Power BI
- Skype

- **Up to 14.6x**
  - Performance gains seen by Microsoft services

- **100s of Millions**
  - # of devices where ONNX Runtime is running

- **Billions**
  - # of requests handled by ONNX Runtime across Microsoft services
Bing QnA - List QnA and Segment QnA
- Two models used for generating answers
- Up to 2.8x perf improvement with ONNX Runtime

Query: empire earth similar games

Games Like Empire Earth
- Total War: Arena.
- Stronghold Kingdoms.
- Rise of Nations.
- Age of Empires 3.
- ... (more items)

19 Games Like Empire Earth - Games Finder
gameslikefinder.com/games-like-empire-earth/

PERFORMANCE
Up to 2.8x perf improvement with ONNX Runtime

<table>
<thead>
<tr>
<th>MAGNITUDE OF IMPROVEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-based</td>
</tr>
<tr>
<td>Transformer w/attention</td>
</tr>
</tbody>
</table>

- ONNX Runtime
- Original framework
ONNX @ Microsoft

Bing Multimedia - Semantic Precise Image Search
- Image Embedding Model - Project image contents into feature vectors for image semantic understanding
- 1.8x perf gain by using ONNX and ONNX Runtime

Query: newspaper printouts to fill in for kids

PERFORMANCE

1.8x perf improvement with ONNX Runtime
Agenda

✓ What is ONNX

✓ ONNX @ Microsoft

❑ What is ONNX Runtime

❑ How to create ONNX models
❖ High performance
❖ Cross platform
❖ Lightweight & modular
❖ Extensible

github.com/microsoft/onnxruntime
ONNX Runtime

• High performance runtime for ONNX models
• Extensible architecture to plug-in optimizers and hardware accelerators
• Supports full ONNX-ML spec (v1.2 and higher, currently up to 1.5)
• Works on Mac, Windows, Linux (ARM too)
• CPU, GPU, Intel edge devices, Nvidia Jeston Nano, ...
• Python, C#, and C APIs
• Code Generation
• Training
ONNX Runtime – Architecture

Graph Optimization
- Node elimination (dropout, identity, etc.)
- Node fusion, constant folding, etc.

Graph Partitioning
- Graph partitioning based on execution providers’ capability
- Greedy algo based on user preferences

Execution Provider
- Plug-in hardware accelerator
- Key APIs
  - GetCapability – given a graph, return a collection of sub-graphs it can run
  - Compile – given a sub-graph (node), return function pointers to run the sub-graph
ONNX Runtime – IR

**Model/Graph/Node**
- In-memory object mapping to ONNX model file format design.
- Offering APIs to read/write a computational graph.

**GraphViewer**
Read-only view of a computational graph. Used in:
- IExecutionProvider (API between Runtime and hardware accelerator)
- Model evaluation (after model optimization and partitioning)
RewriteRule
• An interface created for finding patterns (with specific nodes) and applying rewriting rules against a sub-graph.

GraphTransformer
• An interface created for applying graph transformation with full graph editing capability.

TransformerLevel
• Level 0: Transformers anyway will be applied after graph partitioning (e.g. cast insertion, mem copy insertion)
• Level 1: General transformers not specific to any specific execution provider (e.g. drop out elimination)
• Level 2: Execution provider specific transformers
GraphPartitioner

• Given a mutable graph, graph partitioner assigns graph nodes to each execution provider per their capability and idea goal is to reach best performance in a heterogeneous environment.
• ONNX RUNTIME uses a “greedy” node assignment mechanism
• Users specify a preferred execution provider list in order
• ONNX RUNTIME will go thru the list in order to check each provider’s capability and assign nodes to it if it can run the nodes.

FUTURE:
• Profiling based partitioning
• ML based partitioning
ONNX Runtime – Execution Provider

**IExecutionProvider**
A hardware accelerator interface to query its capability and get corresponding executables.

1) Kernel based execution providers
   These execution providers provide implementations of operators defined in ONNX (e.g. CPUExecutionProvider, CudaExecutionProvider, MKLDNNExecutionProvider, etc.)

2) Runtime based execution providers
   These execution providers may not have implementations with the granularity of ONNX ops, but it can run whole or partial ONNX graph. Say, it can run several ONNX ops (a sub-graph) together with one function it has (e.g. TensorRTExecutionProvider, nGraphExecutionProvider, etc.)

**NodeComputeInfo**
A data structure carries executables returned by runtime-based execution providers.
ONNX Runtime – API Example

```python
import onnxruntime
session = onnxruntime.InferenceSession("mymodel.onnx")
results = session.run([], {"input": input_data})
```

```csharp
using Microsoft.ML.OnnxRuntime;
var session = new InferenceSession("model.onnx");
var results = session.Run(input);
```

...... also available for C
ONNX Runtime – CodeGen

- TVM/Halide Based
- Used by NupharEP and FpgaEP
- NupharEP is open source
ONNX Runtime – Training

• Enables training on Device
  • Helps reinforcement learning, quantized retraining, ...
  • A fast, light-weight runtime with both training and inferencing capability
    • ONNX RT is 3MB binary size, ONNX + Training about 5MB

• Enables large-scale training for multiple frontends and backends
  • A single, unified software stack that
    • Supports multiple training framework frontends (TensorFlow, PyTorch,...)
    • Supports multiple accelerator backends (GPU, ...)
  • A combined SW and HW stack
ONNX Runtime – Training

- Load
- Add Loss Function
- Build Gradient Graph

Diagram:
- Load model
- Add Loss Function: SoftmaxCrossEntropy
- Build Gradient Graph: Loss
Agenda

✓ What is ONNX

✓ ONNX @ Microsoft

✓ What is ONNX Runtime

❑ How to create ONNX models
4 ways to get an ONNX model

- ONNX Model Zoo
- Services like Azure Custom Vision
- Convert existing models
- Train models in systems like Azure Machine Learning service
ONNX Model Zoo: github.com/onnx/models

### Image Classification

This collection of models take images as input, then classifies the major objects in the images into a set of predefined classes.

<table>
<thead>
<tr>
<th>Model Class</th>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet</td>
<td>Sandler et al.</td>
<td>Efficient CNN model for mobile and embedded vision applications. Top-5 error from paper ~10%</td>
</tr>
<tr>
<td>ResNet</td>
<td>He et al., He et al.</td>
<td>Very deep CNN model (up to 152 layers), won the ImageNet Challenge in 2015. Top-5 accuracy: 75.81%</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>iandola et al.</td>
<td>A lightweight model with fewer parameters than VGG. Top-5 accuracy: 94.21%</td>
</tr>
<tr>
<td>VGG</td>
<td>Simonyan et al.</td>
<td>Deep convolutional neural network. Top-5 accuracy: 89.49%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Download</th>
<th>Checksum</th>
<th>Download (with sample test data)</th>
<th>ONNX version</th>
<th>Opset version</th>
<th>Top-1 accuracy (%)</th>
<th>Top-5 accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>44.6 MB</td>
<td>MD5</td>
<td>42.9 MB</td>
<td>1.2.1</td>
<td>7</td>
<td>69.70</td>
<td>89.49</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>83.2 MB</td>
<td>MD5</td>
<td>78.6 MB</td>
<td>1.2.1</td>
<td>7</td>
<td>73.36</td>
<td>91.43</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>97.7 MB</td>
<td>MD5</td>
<td>92.0 MB</td>
<td>1.2.1</td>
<td>7</td>
<td>75.81</td>
<td>92.82</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>170.4 MB</td>
<td>MD5</td>
<td>159.4 MB</td>
<td>1.2.1</td>
<td>7</td>
<td>77.42</td>
<td>93.61</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>230.3 MB</td>
<td>MD5</td>
<td>216.0 MB</td>
<td>1.2.1</td>
<td>7</td>
<td>78.20</td>
<td>94.21</td>
</tr>
</tbody>
</table>
Custom Vision Service: customvision.ai

1. Upload photos and label

2. Train

3. Download ONNX model!
Convert models

- **Tensorflow**: onnx/tensorflow-onnx
- **Keras**: onnx/keras-onnx
- **Scikit-learn**: onnx/sklearn-onnx
- **CoreML**: onnx/onnxmltools
- **LightGBM**: onnx/onnxmltools
- **LibSVM**: onnx/onnxmltools
- **XGBoost**: onnx/onnxmltools
- **SparkML** (alpha): onnx/onnxmltools

Native export
- **Pytorch**
- **CNTK**
from keras.models import load_model
import keras2onnx
import onnx

keras_model = load_model("model.h5")

onnx_model = keras2onnx.convert_keras(keras_model, keras_model.name)
onnx.save_model(onnx_model, 'model.onnx')

import torch
import torch.onnx

model = torch.load("model.pt")

sample_input = torch.randn(1, 3, 224, 224)

onnx_chainer.export(model, sample_input, "model.onnx")
### ONNX & ONNX Runtime - Community Projects

#### Get Involved

<table>
<thead>
<tr>
<th>Discuss</th>
<th>Contribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participate in discussions for advancing the ONNX spec.</td>
<td>Make an impact by contributing feedback, ideas, and code.</td>
</tr>
<tr>
<td>gitter.im/onnx</td>
<td>github.com/onnx</td>
</tr>
<tr>
<td></td>
<td>github.com/microsoft/onnxruntime</td>
</tr>
</tbody>
</table>