All workshop presentations, SIG/WG sessions will be recorded and made available publicly afterwards.
Logistics

- Host of Zoom Meeting will share the slides on screen and record all presentations.
- All participants will be muted except when presenting.
- Questions should be posted in the Slack “onnx-general”
- Please “raise hand” (Zoom feature) if you would like to speak and engage in the discussion.
Goals for the Workshop

● Get the latest updates on ONNX - Processes, Roadmap Releases, and SIGs/WGs
● Learn from the community and how ONNX is being used
● Share feedback on what is working (and what isn’t)
● Learn how to get more involved with ONNX Steering Committee, SIGs and Working Groups
<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:00</td>
<td>Welcome</td>
<td>Sheng Zha (Amazon)</td>
<td>Welcome Logistics</td>
</tr>
<tr>
<td>7:05</td>
<td>ONNX SC Updates</td>
<td>Jacky Chen (Microsoft)</td>
<td>Release 1.8</td>
</tr>
<tr>
<td>7:25</td>
<td>Community Updates</td>
<td>Prasanth Pulavarthi (Microsoft)</td>
<td>Governance</td>
</tr>
<tr>
<td>9:05</td>
<td>Break</td>
<td>Harry Kim (Intel)</td>
<td>Roadmap</td>
</tr>
<tr>
<td>9:15</td>
<td>SIG Updates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:55</td>
<td>Wrap Up</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:00</td>
<td>Welcome</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:05</td>
<td>ONNX SC Updates</td>
<td>Patrick St-Amant (Zetane)</td>
<td>Extract the Maximum Benefits of ONNX to Shorten Your Development Cycle Time and Reduce Guesswork</td>
</tr>
<tr>
<td>7:25</td>
<td>Community Updates</td>
<td>Jianhao Zhang (OneFlow)</td>
<td>ONNX at OneFlow</td>
</tr>
<tr>
<td>9:05</td>
<td>Break</td>
<td>Morgan Funtowicz (Hugging Face)</td>
<td>Efficient Inference of Transformers Models: Collaboration Highlights Between Hugging Face &amp; ONNX Runtime</td>
</tr>
<tr>
<td>9:15</td>
<td>SIG Updates</td>
<td>Danilo Pau (ST Micro)</td>
<td>Flows and Tools to Map ONNX Neural Networks on Micro-controllers</td>
</tr>
<tr>
<td>9:55</td>
<td>Wrap Up</td>
<td>Morgan Funtowicz (Hugging Face)</td>
<td>Efficient Inference of Transformers Models: Collaboration Highlights Between Hugging Face &amp; ONNX Runtime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fabian Bause (Beckhoff Automation)</td>
<td>Neural Automation: Fusion of Automation and Data Science</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faith Xu (Microsoft)</td>
<td>ONNX Runtime Updates: Mobile, Quantization, Training, and More</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jason Knight (OctoML)</td>
<td>Apache TVM and ONNX, What Can ONNX Do for DL Compilers (and vice versa)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alexandre Eichenberger (IBM Research)</td>
<td>ONNX Support in the MLIR Compiler: Approach and Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Matteo Interlandi (Microsoft)</td>
<td>Hummingbird</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neta Zmora (NVIDIA)</td>
<td>Q/DQ is All You Need</td>
</tr>
<tr>
<td>Time</td>
<td>Agenda Item</td>
<td>Presenter(s)</td>
<td>SIG</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------</td>
<td>-------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>7:00</td>
<td>Welcome</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:05</td>
<td>ONNX SC Updates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:25</td>
<td>Community Updates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:05</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:15</td>
<td>SIG Updates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:55</td>
<td>Wrap Up</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ashwini Khade (Microsoft) &amp; Ke Zhang (Alibaba)</td>
<td>Architecture/Infrastructure SIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Michał Karzyński (Intel) &amp; Emad Barsoum (Microsoft)</td>
<td>Operators SIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chin Huang (IBM) &amp; Guenther Schmuelling (Microsoft)</td>
<td>Converters SIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wenbing Li (Microsoft) &amp; Vinitra Swamy (EPFL)</td>
<td>Model Zoo/Tutorials SIG</td>
</tr>
</tbody>
</table>
Engagement & usage (compared to 4/9/20)

- PRs: 1734 (15%↑)
- Contributors: 174 (12%↑)
- Stars: 9.1k (12%↑)
- Dependent Repos: 1773
- Forks: 1.6k (14%↑)
- Papers: 80 (41%↑)
- Models in Zoo: 36 (16%↑)
- Monthly Downloads: 300k
ONNX v1.8 comes with exciting new and enhanced features!

- Windows conda package will be available for the upcoming 1.8 Release (last for v1.1.1)
- Adding Differentiable tags to make Gradient operator better defined
- Remove GraphCall and eliminate the need to implement GraphCall
- Large model (>2GB model) support added for checker and shape_inference
- Graph level shape inference fixes to patch the IR gap introduced since IR version 4
- Node level shape inference fixes for operators
- More operators are supported by version converter
- Add serialization for inputs and outputs of Sequence and Map data types
- Opset 13
  - Extend ControlFlow to allow Sequence type for inputs and outputs
  - Support per-axis scaling for quantizing and dequantizing of tensors
  - Add bfloat16 support

Thank you everyone for your countless hours of work!
ONNX 1.8 Release Schedule

1. Week of Validation (10/13~)
   a. Cut ONNX Release branch
   b. ONNX Release candidate published in PyPI test
   c. Validation in ONNXRuntime
   d. Community validation

2. Week of Release (10/22~): Ready for ONNX 1.8 Release
ONNX

Governance (prasanth)
ONNX open governance update

Steering Committee
https://github.com/onnx/steering-committee

Prasanth Pulavarthi (MS)
Harry Kim (Intel)
Jim Spohrer (IBM)
Sheng Zha (AWS)
Joohoon Lee (Nvidia)

Special Interest Groups (SIGs)
https://github.com/onnx/sigs

Architecture & Infra: Ashwini Khade, Ke Zhang
Operators: Michał Karzyński, Emad Barsoum
Converters: Chin Huang, Guenther Schmuelling
Model Zoo & Tutorials: Wenbing Li

Working Groups (WGs)
https://github.com/onnx/working-groups

Training: Svetlana Levitan
ONNX open governance changes

**Updated licensing:** All code repos under ONNX will be Apache 2. Prior contributions will be reclassified with contributing organization sign-off. Document repos remain CCL.

**CLA -> DCO:**
DCO bot already enabled on all repos under ONNX. Will be made required by 10/19 (already required on main onnx repo). CLA will be turned off once license files updates.

To pass DCO bot, all commits in PRs need to be signed.
Easy to sign: if using command line, `git commit -s`
If using web UI or other tools, include “Signed-off-by: Humpty Dumpty <humpty.dumpty@example.com>” in the commit message (for each commit, not for the PR). Make sure email matches the account you are submitting with.

CONTRIBUTING.md will be updated with tips
ONNX Community Forums

**Gitter** - ONNX rooms will be deprecated by 10/19. Please switch to GitHub Discussions and LF AI Slack

**GitHub Discussions** - new GitHub feature now enabled on onnx/onnx repo, will be enabled on other repos soon. Good for technical questions and discussions that don’t work well as Issues. Issues can be converted to Discussions, but not vice versa.

**Slack** - ONNX channels in LF AI Slack. Channels exist for each SIG and WG. Sign up for LF AI Slack and then join the ONNX channels.
ONNX roadmap discussions

- onnx.ai/roadmap
- Feedback from community

- Impact analysis

- onnx.ai/impact
- Cost analysis

6 weekly community discussions
## Suggested features & their rated impact

<table>
<thead>
<tr>
<th>Operator</th>
<th>Model Zoo</th>
<th>Arch/Infra</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyData alignment (numpy op definitions)</td>
<td>Include quantized models in Zoo</td>
<td>Improve support for large models</td>
</tr>
<tr>
<td>Introduce format (interface and coding style) for op reference implementation</td>
<td></td>
<td>Shape inference (detect error via model checker)</td>
</tr>
<tr>
<td>Simplify function definition</td>
<td></td>
<td>Shape inference (reorg for easier debugging &amp; testing)</td>
</tr>
<tr>
<td>Reduce # of ops</td>
<td>Improve tutorials on Model Zoo</td>
<td>Improve model checker &amp; protobuf loading to prevent sudden termination</td>
</tr>
</tbody>
</table>

- **Med**
  - Improved error handling / exception free

- **High**
  - Reduce # of ops
  - Simplify function definition
  - Expand model test to all models on Model Zoo
  - Include quantized models in Zoo
  - Improve tutorials on Model Zoo
Questions?
Wrap up!
Thank you ...

● Recording of today’s workshop and other applicable content will be shared via ONNX-Announce mailing list when available.
● Please stay engaged and continue to contribute to ONNX and ONNX related projects.
● Remember to use the following ONNX resources:
  ○ Website: https://onnx.ai/
  ○ GitHub: https://github.com/onnx
  ○ Slack: (join https://slack.lfai.foundation - email, password, then find #onnx-general)
  ○ Calendar: https://onnx.ai/calendar
  ○ Mailing List: https://lists.lfai.foundation/g/onnx-announce
ONNX Community Presentations
Extract the maximum benefits of ONNX to shorten your development cycle time and reduce guesswork

Patrick St-Amant
Co-founder and CTO
Zetane Systems
Bio

Co-founder and CTO
Zetane Systems

PhD studies: Category theory and logic
MSc: Foundations of mathematics and computer science
BSc: Mathematics and Physics
Institute for Advanced Study visitor
Zetane Systems

Based in Montreal
Team: software developers, data scientists, simulation, 3D rendering
Recent product release
Industry agnostic

Our software is a 3D engine with a Python API
Does brain imaging, but for artificial neural networks

To know more:
zetane.com
docs.zetane.com
For scientists and developers
Lack of visibility into how algorithms work resulting in wasting valuable time “guessing” how to debug or optimize models.

For subject matter experts
Lack of involvement in the ML process and no understanding of AI algorithms, code, and libraries resulting in lack of trust in what scientists are proposing.

For business leaders
Lack of visibility into how algorithms will impact business, operations and clients in the “real world” and a lack of understanding of risks resulting in slow adoption.

Important problem for industrial adoption

What is Inside the BLACK BOX of Artificial Intelligence?

https://www.analyticsinsight.net/what-is-inside-the-black-box-of-artificial-intelligence
Data flows in ONNX model

Directly from the Python workflow

```python
import zetane as ztn

# Launch the Zetane Engine
zcontext = ztn.Context().launch()

# Create model to send to the engine
zmodel = zcontext.model()

# ONNX
zmodel.onnx(onnx_path).inputs(input_path)

# Update the model and send to the engine
zmodel.update()

# Keras
zmodel.keras(model).update()

# Pytorch
zmodel.torch(torch_model, torch_inputs)
```
## ONNX Model Zoo

### FER+ Emotion Recognition

**Description**

This model is a deep convolutional neural network for emotion recognition in faces.

**Model**

<table>
<thead>
<tr>
<th>Model</th>
<th>Download</th>
<th>Download (with sample test data)</th>
<th>ONNX version</th>
<th>Opset version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion_REPlus</td>
<td>34 MB</td>
<td>31 MB</td>
<td>1.0</td>
<td>2</td>
</tr>
<tr>
<td>Emotion_REPlus</td>
<td>34 MB</td>
<td>31 MB</td>
<td>1.2</td>
<td>7</td>
</tr>
<tr>
<td>Emotion_REPlus</td>
<td>34 MB</td>
<td>31 MB</td>
<td>1.3</td>
<td>8</td>
</tr>
</tbody>
</table>

**Paper**


**Dataset**

The model is trained on the FER+ annotations for the standard Emotion-FER dataset, as described in the above paper.

**Source**

The model is trained ONNX, using the cross entropy training mode. You can find the source code here.

**Demo**

Run Emotion_REPlus in browser - implemented by ONNX with Emotion_REPlus version 1.2

**Inference**

**Input**

The model expects input of the shape `(K, H, W, C)`, where `K` is the batch size.

**Preprocessing**

Given a path `image_path` to the image you would like to score:

```python
import numpy as np
from PIL import Image

def preprocess(image_path):
    img = Image.open(image_path)
    # Perform preprocessing here
    return img
```

https://github.com/onnx/models/tree/master/vision/body_analysis/emotion_ferplus
ONNX graph and API components

Python-Zetane API components:

- Image
- Pointcloud
- 3D meshes
- Numpy
- Video
- Text
- Metric
- Chart
- Panel
- UI elements
Tensors
one-click access, visualizations and statistics
Reduce guesswork and shorten dev time

Focus on the interaction between the data and model

More decisive model improvements

Stop training early

Less wasted time retraining

Increase trust and safety
Autonomous train use case
Inference dashboard

Focus on outliers
xAI dashboard

Original Image

Top 5 Predictions

- 0.73: night snake, Hypsiglena torquata
- 0.26: king snake, lycodon
- 0.21: horned snake, puff adder, sand viper
- 0.01: side-winder, horned rattlesnake, Crotalus cerastes
- 0.00: ring-neck snake, ring-necked snake, ring snake

Model Explainability

LIME

Vanilla Backprop

Grad X Image

Integrated Grads

Grad CAM

Score CAM

Guided Grad CAM

none
Transparent and non-black box ONNX

- Make ONNX tangible and accessible to many
- Create snapshots from the ONNX model zoo
- Help the growth of the ONNX model zoo
- Engage stakeholders
Monitor and debug models in production

- Trigger events
- Record snapshots during inference (or training)
- Inspect snapshots in the Zetane Engine
- React quickly and understand how to improve your models
YOLOv3
We recently launched the Zetane Engine

The community Zetane Viewer is coming soon
Thank you very much!

Q&A

Patrick St-Amant
patrick@zetane.com
https://www.linkedin.com/in/patrick-st-amant

To know more:
  zetane.com
  docs.zetane.com
ONNX at OneFlow

Jianhao Zhang
OneFlow Inc.
About me

- GitHub: daquexian
- A developer at OneFlow Inc.
- An active contributor from ONNX community, a member of operator SIG
- ONNX-related work:
  - ONNX <-> OneFlow conversion (today’s presentation)
  - onnx-simplifier
  - onnx/optimizer
  - resize and softmax op spec
- Also a presenter at ONNX workshop in Shanghai last year (about one of my previous work, integration of Android NNAPI and ONNX Runtime)
What is OneFlow

Oneflow is a brand-new open-source training framework focusing on distributed training. It makes distributed training on multi-machines and multi-devices as simple as on single device.

- Perfectly support container platforms (k8s & docker)
- Handle large models easily
- Almost zero runtime overhead & linear speedup
- Support multiple deep learning compilers (XLA, TensorRT etc)
- Support automatic mixed precision
OneFlow Benchmark

ResNet50v1.5 batch size=128 Throughput images/sec

- OneFlow
- TensorFlow 1.x
- MXNet
- PyTorch
- PaddlePaddle
- TensorFlow 2.x
OneFlow Benchmark

The detailed benchmark report is public at https://github.com/Oneflow-Inc/DLPerf!
“Sounds Great, but..”

Most DL researchers/developers are familiar with TensorFlow/PyTorch/MXNet.

Even though OneFlow is faster, there is a cost to migrate their codecase (mostly the model) to OneFlow.
Solution: Convert TF/PT/MXNet to OneFlow via ONNX
Model to Model Conversion

```python
import torchvision as tv
import oneflow as flow
import oneflow.typing as tp

pytorch_resnet18 = tv.models.resnet18()

@flow.global_function(type="train")
def job(x: tp.Numpy.Placeholder(bs, 3, 224, 224)) -> tp.Numpy:
    y = flow.from_pytorch(pytorch_resnet18, x)
    lr_scheduler = flow.optimizer.CosineScheduler(0.01, 90)
    flow.optimizer.SGD(lr_scheduler).minimize(y)
    return y
```
Model to Model Conversion

```python
import torchvision as tv
import oneflow as flow
import oneflow.typing as tp

pytorch_resnet18 = tv.models.resnet18()

@flow.global_function(type="train")
def job(x: tp.Numpy.Placeholder(bs, 3, 224, 224)) -> tp.Numpy:
    y = flow.from_pytorch(pytorch_resnet18, x)
    lr_scheduler = flow.optimizer.CosineScheduler(0.01, 90)
    flow.optimizer.SGD(lr_scheduler).minimize(y)
    return y
```
Model to Model Conversion

```python
import torchvision as tv
import oneflow as flow
import oneflow.typing as tp

pytorch_resnet18 = tv.models.resnet18()

@flow.global_function(type="train")
def job(x: tp.Numpy.Placeholder(bs, 3, 224, 224)) -> tp.Numpy:
    y = flow.from_pytorch(pytorch_resnet18, x)
    lr_scheduler = flow.optimizer.CosineScheduler(0.01, 90)
    flow.optimizer.SGD(lr_scheduler).minimize(y)
    return y
```
Code to Code conversion (WIP)

# OneFlow code
x2 = flow.layers.conv2d(x1, filters=64,
    kernel_size=7, strides=2, padding=3)
x3 = flow.layer.batch_normalization(x2)
x4 = flow.nn.relu(x3)
x5 = flow.nn.max_pool2d(x4, ksize=3, strides=2, padding=1)
x6 = flow.layers.conv2d(x5, filters=64,
    kernel_size=3, padding=1)

.....

# PyTorch code
import torchvision as tv
model = tv.models.resnet18()
Also, ONNX Helps us Deploy Models on Mobile

As a startup team, we do not have the bandwidth to implement our own mobile inference framework.

Again, ONNX helps us a lot. We convert our model to the existing mobile inference frameworks, like ncnn from Tencent, via ONNX.
Thanks!

Our GitHub: https://github.com/Oneflow-Inc
Efficient inference of transformers models
Collaboration highlights between Hugging Face & ONNX Runtime

Morgan Funtowicz
ML Engineer
Hugging Face OSS

- 25+ employees in 2 offices (NYC & Paris)
- Raised 15 M$ in Serie B

- 😌 transformers (👍 +34,000★)
- 😌 tokenizers (👍 +3,800★)
- 😌 datasets (👍 +4,200★)

- Community model hub with more than 3,000 models
- More than 3To of models stored in the cloud
- More than 5 models uploaded each day
Collaboration with ONNX Runtime

- Looking for a solution to export from PyTorch & TensorFlow.
- Initial integration with transformers to easily export wide variety of our models (25 architectures, from BERT to Reformer & more recently RAG).
- Leverage ONNX Runtime optimizations to speed-up inference on variety of hardwares and platformes.
- Enable quantization for efficient inference.
Collaboration with ONNX Runtime

More information: [Accelerate your NLP pipelines using Hugging Face Transformers and ONNX Runtime](#)
Collaboration with ONNX Runtime

More information: Faster and smaller quantized NLP with Hugging Face and ONNX Runtime
Potential direction

• Integrate data processing operators such as tokenization from our Rust backed tokenizers library

• Supports exporting end-to-end NLP pipelines

• PoC new training features from ONNX Runtime

• PoC for inferencing such models with such various architectures
Conclusion

• ONNX integration has very good perspectives at Hugging Face both for open source projects & internal.

• Well received by the transformers community, especially for the ones looking to put models in production.

• Issues and PRs continues to improve the overall coverage, for instance with recent T5 support.
Flows and Tools to map ONNX Neural Networks on Micro-controllers

Oct 14th 2020

Danilo Pau
Technical Director, IEEE & ST Fellow
System Research and Applications
STMicroelectronics, Agrate Brianza
X-CUBE-AI package as STM32CubeMX cube expansion

STM32CubeMX

Device/Board selector UI → Clock/IP UI configuration → IDE project generator → STM32Cube™ MCU packages → complete IDE project

X-CUBE-AI core
stm32ai (CLI) → C-code generator (optimizer) → Importer → Validation engine → AI filter → AI UI configuration → console UI logger

Optional add-ons apps:
- aiSystemPerformance
- aiValidation

STM32 data base

x-cube-ai app template

- .c
- .h
- .prj
- .a
- .tpl
X-CUBE-AI
User Development Flow

1. Load ANN and filter MCUs
2. Select MCU and create .ioc project
3. Select XCUBEAI Validation
4. Run Analyze
5. Run Validation on x86
6. Run Validation on MCU
   - Either 1-click or via IDE
7. Select and Run System Performance via IDE
8. Generate Application Template
9. Integrate application specific code
10. Run Field Tests
Case Study: ESC-50 (Environmental Sound Classification)

• Dataset
  • 50 classes
  • 40 audio files, 5 sec per class
  • Sampling frequency of recordings: 44.1 KHz
  • Available @ https://github.com/karolpiczak/ESC-50

• Pre processing
  • For each recording, time-frequency spectrogram using 2048 samples windows and 512 samples stride size
  • Transformation of the frequency scale into Mel scale using 128 mel-features
  • Division of the spectrogram into 220ms intervals (128x16 matrix)
  • Ignore low energy spectra whose Frobenius norm is less than 1e-4
  • Normalization respect to maximum energy
Case Study: ESC-50 (Environmental Sound Classification)

ConvNet (Pytorch 1.6.0+cu101)

- Batch size: 100
- Epochs: 200 with early exit
- Optimizer: Adam
- Loss function: Cross Entropy
- Onnx 1.6.0
- RT version 1.1.2
X-CUBE-AI 5.2.0
NUCLEO-STM32H743ZI2, 480MHZ

```
params #          : 150,370 items (587.38 KiB)
macc              : 9,202,672
weights (ro)      : 601,480 B (587.38 KiB)
activations (rw)  : 131,328 B (128.25 KiB)
ram (total)       : 135,624 B (132.45 KiB) = 131,328 + 4,096 + 200
```
X-CUBE-AI 5.2.0
NUCLEO-STM32H743ZI2, 480MHZ

Validation on target

-- Running STM32 C-model - done (elapsed time 9.294s)
-- Running original model
-- Running original model - done (elapsed time 4.900s)

Saving data in "C:\Users\danilo\venv\stm32cube\build\" folder
creating "tinyann_val_m_inputs1.csv" dtype=float32
creating "tinyann_val_m_outputs1.csv" dtype=float32
creating "tinyann_val_m_inputs2.csv" dtype=float32
creating "tinyann_val_m_outputs2.csv" dtype=float32

Cross accuracy report #1 (reference vs C-model)
NOTE: the output of the reference model is used as ground truth/reference value
NOTE: ACC metric is not computed ("-classifier" option can be used to force it)

acc=r.mae=0.021833, mae=0.014533, 122=0.000000
L2r error: 2.78001437e-07

cycles/MACC: 5.72
average for all layers)

109.661 ms
Case Study: Speech Denoise

https://it.mathworks.com/matlabcentral/fileexchange/67296-deep-learning-toolbox-converter-for-onnx-model-format

SPC5-AI v.2.0.0
SPC584B, 120MHZ

params # : 33,125 items (129.39 KiB)
macc : 4,141,181
weights (ro) : 132,500 B (129.39 KiB)
activations (rw) : 16,152 B (15.77 KiB)
ram (total) : 20,796 B (20.31 KiB) = 16,152 + 4,128 + 516

Inference time 348.927 ms
L2r error 8.14623093e-07
How to move forward:

- Needs

- Model zoo of Tiny networks for MCUs trained in Pytorch/Matlab/PaddlePaddle/? exported in ONNX

- Jupyter Notebook tutorials
  - Pytorch Tiny Neural Networks with int8 training aware/post training quantization procedures including exports to ONNX@int8 file format
  - ONNX@fp32 to ONNX@int8 Tiny Neural Networks with post training quantization procedures

- Support of int8 formats: ua/ua, ss/sa, ss/ua
Danilo Pau, graduated at Politecnico di Milano, on 1992 in Electronic Engineering. He joined SGS-THOMSON (now STMicroelectronics) on 1991 and worked on mpeg2 video memory reduction, then video coding, embedded graphics, computer vision, and currently on deep learning. During his career helped in transferring those developments into company products. Also funded and served as 1st Chairman of the STMicroelectronics Technical Staff Italian Community; he is currently Technical Director into System Research and Applications and a Fellow Member of ST. Since 2019 Danilo is an IEEE Fellow, serves as Industry Ambassador coordinator for IEEE Region 8 South Europe, is vice chair of the Task Force on “Intelligent Cyber-Physical Systems” within IEEE CIS and Member for the Machine learning, Deep learning and AI in CE (MDA) Technical Stream Committee IEEE Consumer Electronics Society (CESoc).

Contributed with 113 documents the development of Compact Descriptors for Visual Search (CDVS), CDVS successfully developed ISO-IEC 15938-13 MPEG standard. He was Funding Chair of MPEG Ad Hoc Group on Compact Descriptor for Video Analysis (CDVA), formerly Compact Descriptors for Video Search (CDViS). He also contributes (applications) to MPAI.community recently started by L. Chiariglione. His scientific production consists of 91 papers to date, 78 granted patents and more than 23 invited talks/seminars at various universities and conferences. He was also principal investigator into numerous funded projects at European and Italian level on embedded systems.

Danilo tutored lots of undergraduate students (till Msc graduation), Msc engineers and PhD students from various universities in Italy and India, one of the activities that he likes at most.
Thank you
Neural Automation: Fusion of Automation and Data Science

- Speaker: Fabian Bause
- ONNX Community Virtual Workshop
About Beckhoff and myself

- Fabian Bause
- PhD in Electrical Engineering
- Product Manager TwinCAT at Beckhoff since 01/2016
- Technological responsibilities
  - Machine Learning
  - Integration of MATLAB and Simulink
  - Integration of LabVIEW
  - Signal Processing Libraries
Quick look into the Beckhoff component portfolio

Automation (TwinCAT)

IPC

I/O

Motion
... what is this double cat thing???
The Windows Control And Automation Technology - TwinCAT

PC system running Windows OS

- TwinCAT 3 Engineering Environment based on Visual Studio®
- System Manager
  - Configuration
  - I/O
  - PLC
  - C/C++
  - NC
  - NC I
  - CNC
  - Safety
  - andere
- Programming
  - Non-real-time
  - Real time
  - C#/.NET
  - IEC 61131
  - Object-oriented extension
- TwinCAT transport layer – ADS

Beckhoff IPC running Windows or TC/BSD

- TwinCAT runtime
  - Real-time core
  - TwinCAT 3 runtime
  - PLC
  - TeCOMM
  - Safety
  - NC
  - CNC
  - Simulink®
  - C/C++
  - MATLAB®
  - Simulink® coder
  - andere
  - Third-party programming tools
  - C/C++
- TwinCAT transport layer – ADS
- Fieldbus

Ethernet
What do Beckhoff customers do with all these components? They build machines for...

- Semiconductor Manufacturing
- Medical Engineering
- Energy Industry
- Packaging
- Automotive
- Food Industry
- Warehouse | distribution logistics
- Textile Industry
- Building Automation
Integrating an inference engine into a machine control

**Standard TcCOM in TwinCAT**
- real-time inference engine for ML models
- PLC, C++ and cyclic caller interface
- direct access to EtherCAT slaves, i.e. actuators and sensors
- easy ML model update at runtime

**ONNX support**
- fast growing standardized file format for ML

**Non-blocking parallelization**
- parallel use of one TcCOM object by multiple tasks

**Scalable performance with PC-based control**
- Highly optimized performance by using latest SIMD extensions
Why ONNX is important for Beckhoff?
Why ONNX is important for Beckhoff?

- **ONNX enables seamless workflows**
  - Data Scientists do not need to work into PLC specific languages
  - Automation Engineers and Data Scientists work together while staying in their standard development environment

- **ONNX enables for new business models**
  - Some machine builders establish own Data Science departments, others search for partnerships
  - Maintenance of data driven models during a 20yrs+ runtime of a machine

- **Conjunction of Data Science and Automation is a huge market**
  - Path planning in product transport*, robotics, hand-eye-coordination, …
  - Yield enhancement, RUL prediction, testing, …

*see [www.beckhoff.ai](http://www.beckhoff.ai)
Use Case: MLP for optimized motion planning

TwinCAT Machine Learning

Optimal control

Application
ONNX Runtime 1.5 Release

Highlights: Minimal Builds, Training, Quantization
Minimal Builds

Mobile and Embedded scenarios
ONNX Runtime Minimal Build for Mobile

- Android, iOS, Linux
- X86, ARM
- Same API as existing ORT builds
- Supports all ONNX models
- Model-specific ORT build provides minimal footprint for inferencing on device
- Uses an internal model format to minimize the build size for usage in mobile and embedded scenarios
Convert model to ONNX

Optimization for Mobile

- Optimized model file
  - model.ort
- Operator config file
  - Build onnxruntime pkg

inference on device
Size for ONNX Runtime Mobile

*TfLite package size from: Reduce TensorFlow Lite binary size
†ONNX Runtime full build is 7,546,880 bytes
Training Acceleration

Transformer models
ONNX Runtime Training (Public Preview)

- Seamless integration with existing training frameworks for accelerated training and fine tuning of large transformer models
- Incorporates latest algorithms and techniques such as DeepSpeed/ZeRO and Parasail/Adasum
- Integrates with GPU for distributed execution
## Usage of ORT Training at Microsoft

<table>
<thead>
<tr>
<th>Team</th>
<th>Scenario / Model</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office services</td>
<td>Pre-training TuringNLR</td>
<td>From 4 days to ~2 days (1.4x higher throughput)</td>
</tr>
<tr>
<td>Bing Ads</td>
<td>Pre-training RoBERTa-XL as base model</td>
<td>From 8 days to 4.5 days (1.4x higher throughput)</td>
</tr>
<tr>
<td>Office apps</td>
<td>Fine-tuning GPT-2 for word prediction</td>
<td>Now able to train; stock PyTorch could not train with data parallelism</td>
</tr>
<tr>
<td>Visual Studio</td>
<td>Pre-training GPT-2 Medium for IntelliSense</td>
<td>From 8 days to 6.5 days (1.19x higher throughput)</td>
</tr>
</tbody>
</table>

### Accelerated training with ONNX Runtime

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Days to train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office 365 pre-training 400M+ Model</td>
<td>2.2</td>
</tr>
<tr>
<td>Bing Ads pre-training 500M+ Model</td>
<td>4.5</td>
</tr>
<tr>
<td>Visual Studio fine-tuning 300M+ Model</td>
<td>6.5</td>
</tr>
</tbody>
</table>

- Using PyTorch + ONNX Runtime
- Using PyTorch
Quantization
Latency improvement
Model size reduction

Int8 quantization for 4x reduction in size
Model size reduction

Int8 quantization for 4x reduction in size

Half the size of quantized PyTorch model
Model size reduction

Int8 quantization for 4x reduction in size

Half the size of quantized PyTorch model
…with minimal accuracy tradeoff

Same accuracy as PyTorch

Slightly higher F1 score (precision + recall)
Blog post with more details and E2E Notebook

Faster and smaller quantized NLP with Hugging Face and ONNX Runtime

This post was written by Morsen Funtowicz, Machine Learning Engineer from Hugging Face and Yufeng Li, Senior Software Engineer from Microsoft
Other updates
General
- ONNX 1.7 (opset 12)
- Function expansion support
- Binary Size: Reduced Ops kernel, minimal build for mobile and embedded usage

Performance
- Transformer models (DistilBERT, GPT2, BERT)
- Improved threadpool support for better resource utilization
- Improved performance for inferencing large batch sizes for traditional ML models

APIs and Packages
- IO Bindings
- Allocator sharing between sessions (memory utilization)
- Java API and packages on Maven Central
- NodeJS API
- ARM64 Linux Python package

Windows ML
- UWP apps targeting Windows Store deployment, .NET and .NET Framework applications

Execution Providers
- Select Eps buildable as separate dll (TRT, DNNL, others to come)
- CUDA: 10.2/cuDNN 8.0, CUDA 11 buildable
- TensorRT: 7.1
- OpenVINO: 2020.4
- DirectML: operator coverage and performance improvements, package available on Nuget
- NNAPI: rewritten for broader Android support with more data type and operator coverage, CPU fallback, and improved performance
- AMD MiGraphX: additional data type and operator support, graph optimizations
- ARM NN
- Rockchip NPU
- Xilinx FGPA Vitis-AI
Apache TVM and ONNX
What can ONNX do for DL Compilers (and vice versa)?

Jason Knight - CPO

jknight@octoml.ai
Agenda

… and in 10 minutes …
Let’s go!

Intro to TVM
Cool results (TVM + ONNX)
How does it work?
OctoML’s wishlist for ONNX
An **exploding ecosystem** makes **deployment** painful

Rapidly evolving ML software ecosystem

Cambrian explosion of HW backends
TVM: Bridging the gap as a DL compiler and runtime

Open source, optimization framework for deep learning.

Backends for x86, nVidia/CUDA, AMD, ARM, MIPS, RISC-V, etc

Reduce model time-to-market
Build your model once, run anywhere
Cut capital and operational ML costs

ML-based Optimizations
TVM is an emerging industry standard ML stack

Every “Alexa” wake-up today across all devices uses a model optimized with TVM

“TVM enabled] real-time on mobile CPUs for free...We are excited about the performance TVM achieves.” More than 85x speed-up for speech recognition model.

Bing query understanding: 112ms (Tensorflow) -> 34ms (TVM). QnA bot: 73ms->28ms (CPU), 10.1ms->5.5ms (GPU)

“TVM is key to ML Access on Hexagon” - Jeff Gelharr, VP Technology

Open source 
~428+ contributors from industry and academia.
The power of TVM + ONNX (AKA Results)
Performance: TVM on x86

Performance: TVM on GPU

Performance: TVM on ARM

Comparing TFLite (TF2.1.0) vs. TVM vs. TVM + Autoscheduler on RPI4b CPU

Four core Cortex-A72 @ 1.5GHz fp32 - internal data

Case Study: 50% reduction in Cloud NLP inference costs

Leveraging block sparsity with Apache TVM to halve your cloud bill for NLP

By Joshua Fromm, Bing Xu, Morgan Funtowicz and Jason Knight

2x lower cost on AMD EPYC CPU
Best of both worlds
Not enough time!

- TensorCore performance (better than cuBLAS)
- Classical ML (better than XGBoost and RAPIDS)
- uTVM for TinyML - ML for microcontrollers
- Int{8,4,3,2,1} and posit quantization support
- ML in your browser - WebGPU and WASM as TVM backends
- … and more!
How does it work?
AutoTVM Overview

Automatically adapt to hardware type by learning

\[ \hat{f}(e, c) \rightarrow f(x) \]

\[ c \rightarrow x \]

Expression \rightarrow AutoTVM \rightarrow Code Generator

Search Space \rightarrow Statistical Cost Model

\[ n = \text{te}.\text{var}("n") \]
\[ A = \text{te}.\text{placeholder}((n,), \text{name}="A") \]
\[ B = \text{te}.\text{placeholder}((n,), \text{name}="B") \]
\[ C = \text{te}.\text{compute}(A.\text{shape}, \lambda i: A[i] + B[i], \text{name}="C") \]
\[ \text{print}(\text{type}(C)) \]
ONNX Support in the MLIR Compiler
Approach and Status

Alexandre Eichenberger

Collaborative effort from
IBM Watson/Tokyo Research Labs
and a growing number of external contributors.
Hired in 2001 to help program the IBM/Sony/Toshiba PlayStation 3, I have enjoyed doing research at IBM in Instruction-Level Parallelism, SIMD & thread level parallelism. The last few years, I have worked on supporting OpenMP on our supercomputers, from BG/Q to Coral machines. More recently, I am looking into supporting Deep Neural Networks on a wide range of machines. Just like OpenMP is a great standard to exploit parallelism for a wide range of supercomputers, ONNX is a great standard to support a wide range of frameworks for Deep Neural Networks and related AI tasks.
Multi-Level Intermediate Representation (MLIR)

- **Goals of MLIR.**
  - Significantly reduce the cost of building domain specific compilers.
  - Connect existing compilers together through a shared infrastructure.
  - Part of LLVM compiler & governance.
Architecture of ONNX-MLIR Compiler

- Consumes ONNX model and produce inference executables using 2 new dialects:
  - ONNX: representation of native ONNX operations,
  - KRNL: representation to lower ONNX to loops.
Integration of ONNX Specs within MLIR Framework

• **Ingest ONNX Specs directly into MLIR.**
  • Script transforms ONNX Specs into LLVM TableGen format.
  • Describes ONNX operations for MLIR (inputs, attributes, outputs, types).
  • Drives validation and shape inference.
Pattern-Matching Transformations in MLIR

- High level description of ONNX to ONNX transformations.
  - E.g MatMul and Add into a GEMM operation.

\[
\begin{align*}
%0 &= \text{"onnx.MatMul"}(\%a0, \%a1) : \text{tensor<10x10xf32>, tensor<10x10xf32>} \rightarrow \text{tensor<10x10xf32>} \\
%1 &= \text{"onnx.Add"}(\%0, \%a2) : \text{tensor<10x10xf32>, tensor} \\
%0 &= \text{"onnx.Gemm"}(\%a0, \%a1, \%a2) \\
&\quad \{ \text{alpha} = 1.000000e+00, \text{beta} = 1.000000e+00, \text{transA} = 0, \text{transB} = 0 \} : \\
&\quad \{ \text{tensor<10x10xf32>, tensor<10x10xf32>, tensor<10x10xf32>} \rightarrow \text{tensor<10x10xf32>} \}
\end{align*}
\]
Example: MNIST model to ONNX dialect

ONNX Model

```
func @main_graph(%arg0: tensor<1x1x28x28xf32>, %arg1: tensor<8x1x5x5xf32>,
                   %arg2: tensor<8x1x1xf32>, %arg3: tensor<16x8x5x5xf32>,
                   %arg4: tensor<16x1x1xf32>, %arg5: tensor<2xi64>, %arg6: tensor<16x4x4x10xf32>, %arg7: tensor<2xi64>,
                   %arg8: tensor<1x10xf32>) -> tensor<1x10xf32> {
  %0 = "onnx.Constant"() { sparse_value = [], value = [256, 10]} : () -> tensor<2xi64>
  %1 = "onnx.Reshape"(%arg6, %0) : (tensor<16x4x4x10xf32>, tensor<2xi64>) -> tensor<256x10xf32>
  %2 = "onnx.Conv"( %arg0, %arg1, null)
    { auto_pad = "NOT_SET", group = 1, kernel_shape = [5, 5], strides = [1, 1] dilations = [1, 1] } : 
      ( tensor<1x1x28x28xf32>, tensor<8x1x5x5xf32> ) -> tensor<1x8x28x28xf32>
  %3 = "onnx.Add"(%2, %arg2) : (tensor<1x8x28x28xf32>, tensor<8x1x1xf32>) -> tensor<1x8x28x28xf32>
  [...]
}
```

- **Output representation:**
  - all shapes inferred, propagated, and verified,
  - all parameters verified and normalized.

```
ONNX
--EmitONNXIR
mnist.onnx
```
Motivation for KRNL Dialect

- Dialect used to lower ONNX operations to loop code.
- Designed for customizable optimizations such as tiling, fusion, parallelization.
  - It is hard to revert an optimization, except if...
  - Instead of performing the optimization, we just record it (i.e. build a recipe of optimizations).
  - And can alter it later as needed by other optimization.

- Example:
  - Try to fuse 2 kernels that were tiled by different amount:

  - Make them compatible by simply edit B’s recipe from [2, 2] -> [5, 5]
  - Greater freedom and flexibility for compilers to traverse the schedule space!
ONNX-MLIR Runtime Interface

**C API.**
- invoked using `run_main_graph` function.

```c
// Create input tensors:
OMTensor *x1 = omTensorCreate(...);
OMTensor *x2 = omTensorCreate(...);

// Create input tensor list:
OMTensor *list[2] = {x1, x2};
OMTensorList *input = omTensorListCreate(list, 2);

// Invoke inference function & get prediction.
OMTensorList *outputList = run_main_graph(input);
OMTensor *y = omTensorListGet0mtByIndex(outputList, 0);
```

**Python API.**
- use execution session,
- input/output with numpy,
- thanks @gperrotta.

```python
from PyRuntime import ExecutionSession

# Construct execution session using compiled model file path
# and inference function symbol name, default is "run_main_graph".
session = ExecutionSession("LeNet.so", "run_main_graph")

# Specify input, run model and retrieve output.
input = np.array(...)
outputs = session.run(input)
prediction = outputs[0]
```
Get Involved

• Big thanks.
  • to the 15+ external contributors.

• Learn more.
  • Code: [https://github.com/onnx/onnx-mlir](https://github.com/onnx/onnx-mlir)
  • Additional documentation: [http://onnx.ai/onnx-mlir](http://onnx.ai/onnx-mlir)

• Status.
  • Support 50+ commonly used ONNX operations.
  • Can compile mnist, resnet to LLVM.

• Actively prototyping.
  • ONNX, KRNL & buffer optimizations.
  • ONNX-ML support.
Auto-scheduling Overview

Widens search space even further than AutoTVM 1.0
OctoML’s wishlist for ONNX
We wish ONNX had...

- Even broader op coverage (e.g., EmbeddingBag)
- Broader non-ML (but adjacent) support:
  - More classical ML
  - GCNN/DGL
  - Graph workloads (GraphBLAS, Metagraph)

And on the “pie-in-the-sky” list:

- Framework integrations
  - PyTorch: so we don’t have to deal with torchscript
  - MLIR dialect so we can easily plug into TensorFlow (for runtime JIT)

- Quantization-aware standardization
  - For eg: canonicalization of models coming out of Quantization-aware-training pipelines

Thanks!
Compiling Traditional ML Pipelines into Tensor Computations for Unified Machine Learning Prediction Serving

Matteo Interlandi, Karla Saur, Supun Nakandala, Gyeong-In Yu, Markus Weimer, Konstantinos Karanasos, Carlo Curino
Outline

• Motivate why model prediction for Traditional ML is an important problem

• Briefly introduce how classical models can be compiled into tensor operations

• Project status
Motivation

Specialized Systems have been developed (mostly focus on neural networks)

Support for traditional ML methods is largely overlooked (widely used in practice because state of the art on tabular data)
Traditional ML Models


Data Science throw the looking glass: [https://arxiv.org/abs/1912.09536](https://arxiv.org/abs/1912.09536)
Hummingbird

A compiler translating traditional ML models into tensor computations for unified ML prediction serving

Benefits:
(1) Exploit the already available DNN runtimes
(2) Exploit current (and future DNN) optimizations
(3) Seamless hardware acceleration
(4) Significant reduction in engineering effort
Traditional ML Operators

- Traditional ML models are composed by: **featurizers** and **ML models**

- Each featurizer is defined by an **algorithm**
  - e.g., compute the one-hot encoded version of the input feature

- Each trained model is defined by a **prediction** function
  - Prediction functions can be either **algebraic** (e.g., linear regression) or **algorithmic** (e.g., decision tree models)
  - Algebraic models are easy to translate: just implement the same formula in tensor algebra!
Translating Trees

\[ \mathbf{x} \in \mathbb{R}^6 \]

- Internal node: \( n_* \)
- Leaf node: \( l_* \)

Diagram:

```
      x \in \mathbb{R}^6
     /   \\
   n_1   n_2
  /   \   /   \\
 n_3   T  n_4   F
 / \   /   / \   /   / \\
x_1 > 1.8 x_3 > 5.1 x_5 > 0.4
 /     /     /     / \\
 l_1   n_3   l_4   l_5
 / \   /   /     /     \\
x_3 > 2.4 1 4 5 \\
 / \   /   / \\
 l_2   l_3   0 0 \\
      2 3 0 0 
```
Translating Trees

\[ x \in \mathbb{R}^6 \]

\[ n_1 \quad x_3 > 5.1 \]

\[ n_2 \quad x_1 > 1.8 \]

\[ n_3 \quad x_3 > 2.4 \]

\[ n_4 \quad x_5 > 0.4 \]

\[ n_5 \]

\[ l_1 \]

\[ l_2 \]

\[ l_3 \]

\[ l_4 \]

\[ l_5 \]

\[ b = \begin{pmatrix} -5.1 \\ -1.8 \\ -2.4 \\ -0.4 \end{pmatrix} \]

\[ n_*, l_* \in [0,1] \]

\[ lv = \begin{pmatrix} 10 \\ 20 \\ \vdots \\ 50 \end{pmatrix} \]
Translating Trees

Evaluate all conditions together

\[
\begin{align*}
n_1 & \iff x_3 > 5.1 \\
n_2 & \iff x_1 > 1.8 \\
n_3 & \iff x_3 > 2.4 \\
n_4 & \iff x_5 > 0.4
\end{align*}
\]

\[b = \begin{pmatrix}
-5.1 \\
-1.8 \\
-2.4 \\
-0.4 \\
\end{pmatrix}
\]

\[lv = \begin{pmatrix}
10 \\
20 \\
\vdots \\
50 \\
\end{pmatrix}
\]
Translating Trees

Evaluate all conditions together

Evaluate all paths together

$x \in \mathbb{R}^6$

$n_1 \quad x_3 > 5.1$

$n_2 \quad x_1 > 1.8$

$n_4 \quad x_5 > 0.4$

$n_3 \quad x_3 > 2.4$

$n_5 \quad x_5 > 0.4$

$l_1 \quad 1$

$l_2 \quad 2$

$l_3 \quad 0$

$l_4 \quad 4$

$l_5 \quad 5$

$b = \begin{pmatrix} -5.1 \\ -1.8 \\ -2.4 \\ -0.4 \end{pmatrix}$

$l_*, n_* \in [0,1]$
Translating Trees

• Random forest, boosting, ...

[Diagram showing trees and neural network with output nodes]
Hummingbird: Status

• Open sourced in May: [https://aka.ms/hb-code](https://aka.ms/hb-code) (See also: Blog Paper Demo)
  • Integration with ONNX converters (LightGBM): Blog
  • Hummingbird is part of the PyTorch Ecosystem
  • Paper will be presented at OSDI 2020

1.7K GitHub stars

136 GitHub forks

>20 external PRs
(≈5 regular/repeat contributors, 10 total external contributors),
20 issues filed by external users

6 user-created blog posts and a video tutorial with >1k views
### Future work: Integration with other ONNX converters

<table>
<thead>
<tr>
<th>Operator Group</th>
<th>Supported Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Classifiers</td>
<td>Logistic Regression, Linear SVC, SVC, NuSVC, SGDClassifier, LogisticRegressionCV</td>
</tr>
<tr>
<td>Tree Methods</td>
<td>DecisionTreeClassifier/Regressor, RandomForestClassifier/Regressor, GradientBoostingClassifier/Regressor, ExtraTreesClassifier/Regressor, XGBClassifier/Regressor, LGBMClassifier/Regressor/Ranker</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>MLPClassifier</td>
</tr>
<tr>
<td>Others</td>
<td>BernouliNB, KMeans</td>
</tr>
<tr>
<td>Feature Selectors</td>
<td>SelectKBest</td>
</tr>
<tr>
<td>Decomposition</td>
<td>PCA, TruncatedSVD</td>
</tr>
<tr>
<td>Feature Pre-Processing</td>
<td>SimpleImputer, Imputer, ColumnTransformer, RobustScaler, MaxAbsScaler, MinMaxScaler, StandardScaler, Binarizer, KBinsDiscretizer, Normalizer, PolynomialFeatures, OneHotEncoder, LabelEncoder, FeatureHasher</td>
</tr>
<tr>
<td>Text Feature Extractor</td>
<td>CountVectorizer</td>
</tr>
</tbody>
</table>
Thank you!

hummingbird-dev@microsoft.com
Q/DQ IS ALL YOU NEED

Neta Zmora,
Oct 14, 2020
AGENDA

Q/DQ are necessary and sufficient

How do we optimize graphs with Q/DQ?

Where do we insert Q/DQ in our graphs?
Q/DQ ARE NECESSARY

- For quantizing network input; and dequantizing network output.
- For changing precision mid-graph (for example: input to softmax).
Fake quantization is the prevailing approach used for DNN quantization.

Forward pass: \( \hat{x} = \text{dequantize}(\text{quantize}(x)) \)

ONNX QuantizeLinear and DequantizeLinear naturally represent fake quantization.
Minimize activation bandwidth

Original Graph

Graph after Q/DQ migration

*In this presentation we assume per-tensor quantization for activations, and per-channel quantization for weights. This produces the best results and simplifies the math.
Commuting with Scale/Shift:

- OP -> Q =? Q -> OP
- DQ -> OP =? OP -> DQ
Fusion Opportunities

MatMul Fusion Reordering
Consider a scalar multiplication:

\[ c = a \cdot b \]

Suppose fake quantization nodes are attached to the input:

\[
\begin{align*}
  c = \hat{c} \\
  \hat{c} &= qdq(a)qdq(b) \\
  &= ((a_q - z_a)s_e)((b_q - z_q)s_b) \\
  &= s_a s_b (a_q b_q - (z_b a_q + z_a b_q - z_a z_b))
\end{align*}
\]

With real quantization we would compute:

\[
\hat{c} = dq(c_q) = dq(a_q b_q) = dq(q(a)q(b))
\]

Which is equivalent to the fake quantized expression:

\[ dq(c_q) = (c_q - z_c)s_c \]

where

\[
\begin{align*}
  s_c &= s_a s_b \\
  z_c &= z_a a_q + z_q b_q - z_a z_b
\end{align*}
\]

The only difference between the original graph and the rewritten graph is the order of operations.
Q/DQ PLACEMENT RECOMMENDATIONS

- Quantize all inputs of linear operations, by inserting fake quantization (Q/DQ) in front of them.
- By default, don’t quantize operator outputs.
- Be conservative when adding Q/DQ nodes.
- Use per-tensor quantization for activations; and per-channel quantization for weights.
Wish we had more time!
Title: Micro-service Generation for ONNX model in Acumos

Abstract: In the same way as models developed in Python, R, C++ and Java, ONNX models can now take benefits of all the Acumos functionalities and most particularly they can be dockerised and transformed as a micro-service to ease their deployment. During this time slot I will explain briefly how we succeeded to do that and what is the future of our Acumos ONNX onboarding client.

Bio: Philippe Dooze (Orange) joined Orange Labs R&D in 2010, and he mainly works on projects involved in network QoS based on data and big data analysis. He joined Acumos LF AI project in 2018 as a Project Team Leader of on-boarding component, Three month ago, he became Project Team leader of Model Management component that groups on-boarding, micro service generation and model deployment.
Architecture & Infra SIG Update

ONNX Workshop October 2020

Ashwini Khade, Microsoft
Ke Zhang, Alibaba
Updates and Announcements

- ONNX optimizers moved to separate repo: [https://github.com/onnx/optimizer](https://github.com/onnx/optimizer)
- CI Updates
  - CI improvements for improved reliability
  - Moved to AzurePipelines to speed up the runs
- Shape Inference
  - Numerous improvements and bug fixes to node level shape inference
  - Updates to graph level shape inference (to path IR gap introduced since IR version 3)
- ONNX Checker
  - Updates to improve model validation coverage
  - Limited support for large models
- ONNX Package Updates
  - Windows Conda package fixed (was broken since version 1.1)
  - Linux Manylinux image updated to 2014
- Version Converters updates
Upcoming Investments

- Reduce ONNX package size
- Remove Optimizes from onnx package
- Infra support for reference implementation (explore plugging reference implementation as pyOp for onnxruntime)
- Continue investments in shape inference, onnx checker and CIs
- onnx.ai/impact
Get Involved!

- Slack Channel: https://slack.lfai.foundation and join onnx-archinfra
- Meetings and announcements are on slack channel
- Arch Infra SIG meeting will be bimonthly here onwards (announcement on slack channel)
- Submit and review PRs
- Participate in discussions on slack and on github
AGENDA

- Operators SIG
- Add new operator update
- Proposal / improvement
- Discussion: Version converter
- Discussion: PR and Issues
### GOAL

<table>
<thead>
<tr>
<th>Keep Up</th>
<th>Quality</th>
<th>Clarity</th>
<th>Size</th>
<th>PRs and Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep up with the latest progress in AI</td>
<td>Improve the quality of ONNX Operators</td>
<td>Reduce ambiguity</td>
<td>Avoid bloating ONNX spec</td>
<td>Keep up with PRs and operator issues</td>
</tr>
</tbody>
</table>
PARTICIPANTS

- Akinlawon Solomon (Qualcomm)
- Darren Crews (Intel)
- Dilip Sequeira (NVidia)
- Ganesan Ramalingam (Microsoft)
- Itay Hubara (Habana)
- Jianhao Zhang (JD)
- Ke Zhang (Alibaba)
- Leonid Goldgeisser (Habana)
- Milan Oijaca (Qualcomm)
- Ofer Rosenberg (Qualcomm)
- Rajeev K Nalawadi (Intel)
- Scott Cyphers (Intel)
- Shlomo Raikin (Habana)
- Simon Long (GraphCore)
- Spandan Tiwari (Microsoft)
- Wei-Sheng Chin (Microsoft)
- Weiming Zhao (Alibaba)
- Liqun Fu (Microsoft)
COMMUNICATION

- Slack channel: https://slack.lfai.foundation and join onnx-operators
- Discussions on GitHub PRs and issues
- Meetings announcement are on Slack and Gitter
- Docs and meeting notes are in onnx/sigs https://github.com/onnx/sigs/tree/master/operators
- Deprecated: Gitter channel: https://getter.im/onnx/operators
ADDING NEW OPERATOR ISSUE

- Reference implementation in Python isn’t enough.
- Runtime and framework writers, start implementing new operators close or after ONNX release.
  - Any issue cause delay to the release.
  - Or worse, cause a patch release.
- Operator behavior might not match existing framework, especially in corner cases.
ADDING NEW OPERATOR UPDATE

- Unit tests need to have the same coverage as the original framework.
- Test data need to be generated from the original framework to match behavior.
- [Optional] verify the new operator/function in a runtime/framework that support ONNX.
PROPOSAL

Feel free to propose any improvements, such as:

- Better testing, validation and coverage of ONNX operators.
- Better documentation generation.
- More operators.
- A lot of existing manual steps need automation.

For any big proposal, you will be invited in the SIG meeting to present it.
GET INVOLVED:
SUBMIT AND REVIEW PRS
PR REVIEW

- PRs should be marked with the Operator label
  - [https://github.com/onnx/onnx/pulls?q=is:pr+is:open+label:operator](https://github.com/onnx/onnx/pulls?q=is:pr+is:open+label:operator)
- Ops Contributors Group should review the PRs according to guidelines
- Mature PRs can be discussed during bi-weekly sync
- Final approval by member of SIG-operators-approvers group
GITHUB DISCUSSION ISSUES

- Many open discussions marked with Operator label
- Ops Contributors Group should be active in discussions and encourage submission of PRs
- We should decide which discussions can be closed
- Still looking for best way to triage this large number of open issues
VERSION CONVERTER

- Used to convert from one OPSet to another or vice versa.
- Currently, it is outdated.
- Should we enforce every operator PR to update the version converter?
TIME MAJOR FLAG FOR RECURRENT

- Issue and PR:
  - https://github.com/onnx/onnx/issues/2159
  - https://github.com/onnx/onnx/pull/2922
  - https://github.com/onnx/onnx/pull/2284

- Current recurrent operator in ONNX, operate on:
  - [seq_length, batch_size, input_size]

- Most framework support:
  - [seq_length, batch_size, input_size] and [batch_size, seq_length, input_size]
  - Some framework hide the batch axis.
THANKS FOR COMING!!!

Operator SIG resources

- Slack channel: [https://slack.lfai.foundation](https://slack.lfai.foundation) and join onnx-operators
- Documents and artifacts: [https://github.com/onnx/sigs/tree/master/operators](https://github.com/onnx/sigs/tree/master/operators)
Converters SIG Updates

ONNX Workshop 10/14/2020

Chin Huang, IBM
Guenther Schmuelling, Microsoft
Kevin Chen, Nvidia
Converters SIG Updates

- Converters updates
  - Frontend converters
  - Backend converters

- General discussions
  - What we would like to see...
  - What we would like to clarify...
  - Interesting operators
Frontend Converter Updates - keras2onnx

- Opset 12 fully supported in keras2onnx v1.7.0
- Support for most huggingface/transformers models
- Improved RNN model conversion
- Validated successful conversion for 120+ models in github
- Bug fixes
Frontend Converter Updates - pytorch exporter

- 15+ new torch operators supported for export.
- Support for ONNX Opset 12.
- Support for large models (> 2GB protobuf limit), including large attributes.
- Enhanced support for all TorchVision models, including dynamic input size export.
- Improved custom op export experience.
- Export support for torch.FakeQuantize to support basic QAT workflow.
- Several updates to existing ops and optimizations.
- On the roadmap:
  - Improvements to ScriptModule export
  - Opset 13 support
Frontend Converter Updates - tf2onnx

• Fixes for tf-2.x, tested up to tf-2.3
• Support for models > 2GB (--large_model)
• Support for quantization aware training (using QDQ)
• Improvements to optimizer pass
• Constant folding for almost all TF ops
• More fusing of nodes, ie. Batchnorm with Conv
• Major improvements for conversion speed on large models (3 to 5 times faster)
• Bug fixes
• Currently supported: tensorflow: up to tf-1.15, tf-2.3 | onnx: opset-7 - opset-12 | python 3.6-3.8

We are now working on improving out of the box conversion rates.
Backend Converter Updates – ONNX-TensorRT

TensorRT 7.2.1
- Support for parsing models with external data
- New API for interfacing with TensorRT’s refit feature
- New tooling (link)
  - **ONNX-GS** - Custom wrapper around the existing ONNX python API for easier creation and modification of ONNX graphs
  - **Polygraphy** - Toolkit that allows running and debugging DL models between different backends.

Future plans
- Continuously improve operator support
- Work more closely with front-end converters
Backend Converter Updates – ONNX-TF

- Tensorflow 2.0 native support, export as saved model, instead of graph pb
- User choice of target device, CPU vs GPU, for optimized graph
- Auto and user input for data type cast
- Supported Tensorflow versions: 1.15 and 2.3
- ONNX opset 12 support
- Upcoming
  - ONNX opset 13 support
  - Investigation in training and onnx-ml
General discussions

What we would like to see…

- Custom ops best practices: 1. convert to backend framework models 2. execute in backend runtime
- Consistency between op schema, doc, and unit tests
- Use of checker to ensure model quality and integrity in frontend and backend
- Working standard backend tests
  - Sequence as an input type
  - New data type in opset 13: bfloat16
- ONNX-ML reference implementation and unit tests at operator/node level, and more ML models in model zoo
- Visualization of subgraphs (loop, if, scan): helpful onnx runtime tool dump_subgraphs.py,
  https://github.com/microsoft/onnxruntime/blob/master/tools/python/dump_subgraphs.py
General discussions

What we would like to clarify…

- Inference accuracy could be slightly off between backend frameworks
- Optional inputs with default values or blank input names?
- Move between attributes and inputs might cause issues for backend frameworks
- Model training use cases specifically for frontend and backend converters, APIs, models in model zoo
General discussions

Operators we discussed
  • Resize (variants difficult to understand and execute, if not impossible, leading to partial support from framework converters)
  • Loop (and nested loops, might need access to initializers out of loop scope)
  • Clip (optional input with blank input name in unit test)
  • OneHot, NonMaxSuppression (depth input is documented as a scalar, which could be a scalar or 1-d tensor of size 1 in schema and unit tests)
  • SplitToSequence (split input could be a scalar or a 1-d tensor)
  • NegativeLogLikelihoodLoss, SoftmaxCrossEntropyLoss (no direct mappings in some frameworks)
Thank You and Join our Slack and Meetings

#onnx-converters

https://lists.lfai.foundation/g/onnx-sig-converters/
ONNX Model Zoo + Tutorials SIG Update

Wenbing Li, Microsoft
Vinitra Swamy, EPFL
10/14/2020
The ONNX Model Zoo is a collection of pre-trained, state-of-the-art model set.

Model Zoo CI is active! (onnx#307)

- Running the ONNX Checker on each new model, working towards ONNX Runtime testing on model inputs / outputs

New and Updated Models

- EfficientNet-Lite 4 (onnx#324)
- YOLO V4 (onnx#322)
- roBERTa (onnx#338)
- T5 (onnx#357)
- SSD MobileNet v1 (onnx#328)
- RetinaNet (onnx#308)
- ShuffleNet (onnx#250)
- Updates to SuperResolution, GoogLeNet, GPT-2, SqueezeNet, MNIST

Git LFS Migration complete! (onnx#271)

- Download all models in the zoo with one command
  - git lfs pull --include="*" --exclude=""
- All models are stored within the model zoo for long-term storage

The ONNX Model Zoo is a collection of pre-trained, state-of-the-art model set.
1. More than 50+ models
2. Contributed by 40+ community members
3. 10+ categories (like Image classification, object detection, LM etc.)
4. 2.7K Stargazers

Key numbers:
- Image classification
- Object detection and image segmentation
- Body, face and gesture analysis
- Image manipulation
- Machine comprehension
Most popular models: Mobilenet, ResNet, YoloV4, ArcFace
Most popular tutorials: Tensorflow->ONNX, Visualizing an ONNX model, Pytorch->ONNX, ONNX RT server with SSD
Vinitra Swamy has recently left Microsoft to begin her PhD in Switzerland (at EPFL). She is stepping down as SIG lead but will still be an active member in the model zoo efforts.

Wenbing Li is a core contributor to the ONNX converter efforts and is taking over leadership of the Model Zoo + Tutorials SIG.

More models for mobile/embedded scenarios

Quantized model

State of art models

ONNX opset upgrading

The coming hot topics
Files needed for PR
ONNX Model File
Test input data
Inference example/ tutorial if applicable
ReadME.md

Model verification
onnx.checker
Inference checker by the test data

Contribute your models
Join us!
- Slack Channel: https://slack.lfai.foundation and join onnx-modelzoo
- Monthly meetup
- Info page: https://github.com/onnx/sigs/tree/master/models-tutorials

Model Zoo SIG Resources