ONNX | Workshop 10/14/2020

ONNX

Welcome!

Disclaimer

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

Agenda -1

7:00	Welcome
7:05	ONNX SC Updates
7:25	Community Updates
9:05	Break
9:15	SIG Updates
9:55	Wrap Up

Sheng Zha (Amazon)	Welcome Logistics Goals Agenda State of the State: ONNX Growth
Jacky Chen (Microsoft)	Release 1.8
Prasanth Pulavarthi (Microsoft)	Governance
Harry Kim (Intel)	Roadmap

Agenda - 2

7:00	Welcome
7:05	ONNX SC Updates
7:25	Community Updates
9:05	Break
9:15	SIG Updates
9:55	Wrap Up

Patrick St-Amant (Zetane)	Extract the Maximum Benefits of ONNX to Shorten Your Development Cycle Time and Reduce Guesswork
Jianhao Zhang (OneFlow)	ONNX at OneFlow
Morgan Funtowicz (Hugging Face)	Efficient Inference of Transformers Models: Collaboration Highlights Between Hugging Face & ONNX Runtime
Danilo Pau (ST Micro)	Flows and Tools to Map ONNX Neural Networks on Micro-controllers
Fabian Bause (Beckhoff Automation)	Neural Automation: Fusion of Automation and Data Science
Faith Xu (Microsoft)	ONNX Runtime Updates: Mobile, Quantization, Training, and More
Jason Knight (OctoML)	Apache TVM and ONNX, What Can ONNX Do for DL Compilers (and vice versa)
Alexandre Eichenberger (IBM Research)	ONNX Support in the MLIR Compiler: Approach and Status
Matteo Interlandi (Microsoft)	Hummingbird
Neta Zmora (NVIDIA)	Q/DQ is All You Need

Agenda - 3

7:00	Welcome
7:05	ONNX SC Updates
7:25	Community Updates
9:05	Break
9:15	SIG Updates
9:55	Wrap Up

Ashwini Khade (Microsoft) & Ke Zhang (Alibaba)	Architecture/Infrastructure SIG
Michał Karzyński (Intel) & Emad Barsoum (Microsoft)	Operators SIG
Chin Huang (IBM) & Guenther Schmuelling (Microsoft)	Converters SIG
Wenbing Li (Microsoft) & Vinitra Swamy (EPFL)	Model Zoo/Tutorials SIG

ONNX

State of the state

Engagement & usage (compared to 4/9/20)

15%↑

1734

PRs

12%↑

174

Contributors

12%↑

9.1k

Stars

1773

Dependent Repos

14%↑

1.6k

Forks

41%↑

80

Papers

16%↑

36

Models in Zoo

300k

Monthly Downloads

Support

Creation/ Manipulation

Caffe2 · Chainer · Cognitive Toolkit · LibSVM · mxnet · ML.NET · PyTorch · Keras · Tencent · [M]^s · scikit learn · dmlc XGBoost · PaddlePaddle · ML · sas · DeepCore deepcore.io · MATLAB · Neural Network Libraries · SIEMENS

NEW

NeoML · MyCaffe™ · Yandex CatBoost · Tengine · Apache SINGA

Run/ Compile

ONNX RUNTIME · intel AI · Qualcomm · SYNOPSYS · CEVA · Tencent · Windows · NVIDIA · habana · nGraph · skymizer · SOPHON · MACE Mobile AI Compute Engine · NNOIR · GRAPHCORE · Rockchip · vespa · tvm

NEW

BECKHOFF TwinCAT 3 · cadence · tensilica · Acumos

Visualization/ Test Tools

NETRON · Visual DL · Sionnx · ZETANE

Coming soon: ONNX 1.8 (Release mgr: Jacky Chen)

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

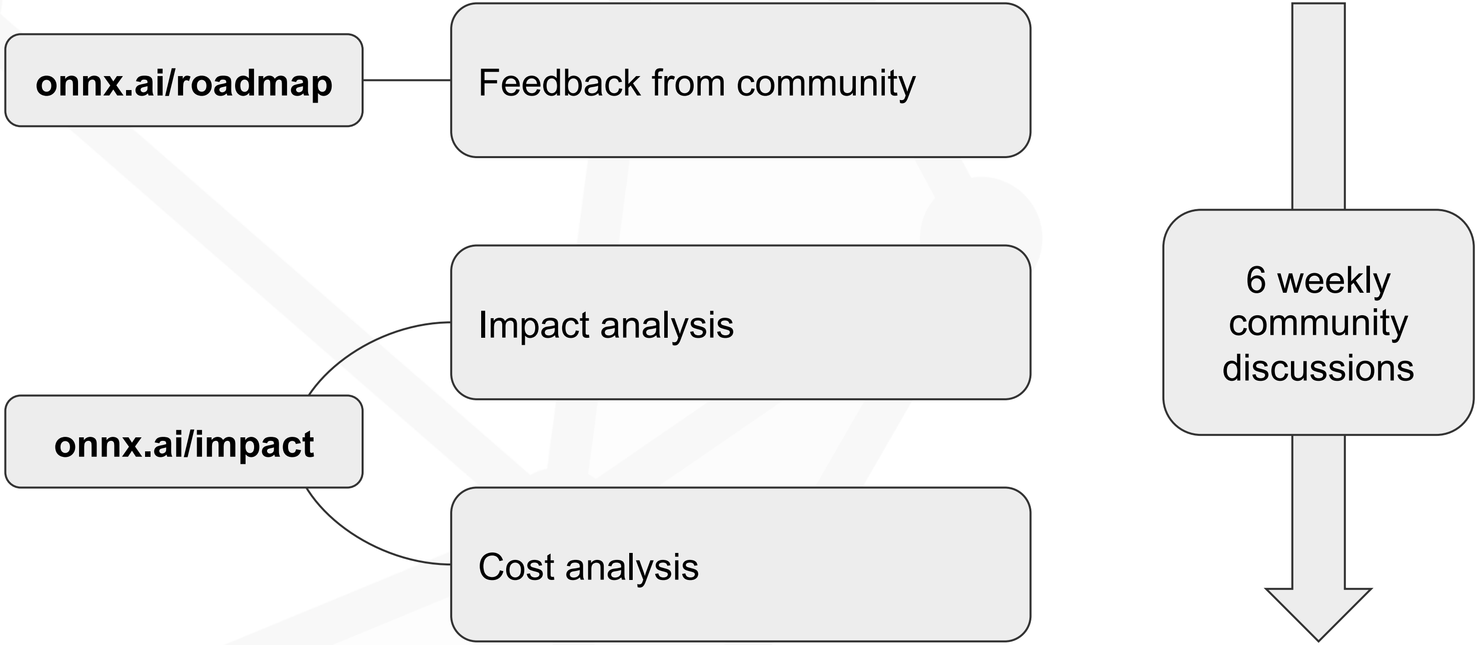
GitHub Discussions - new GitHub feature now enabled on onnx/onnx repo, will be enabled on other repos soon.
Good for technical questions and discussions that don’t work well as Issues.
Issues can be converted to Discussions, but not vice versa.

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 (harry)

ONNX roadmap discussions



onnx.ai/roadmap

Feedback from community

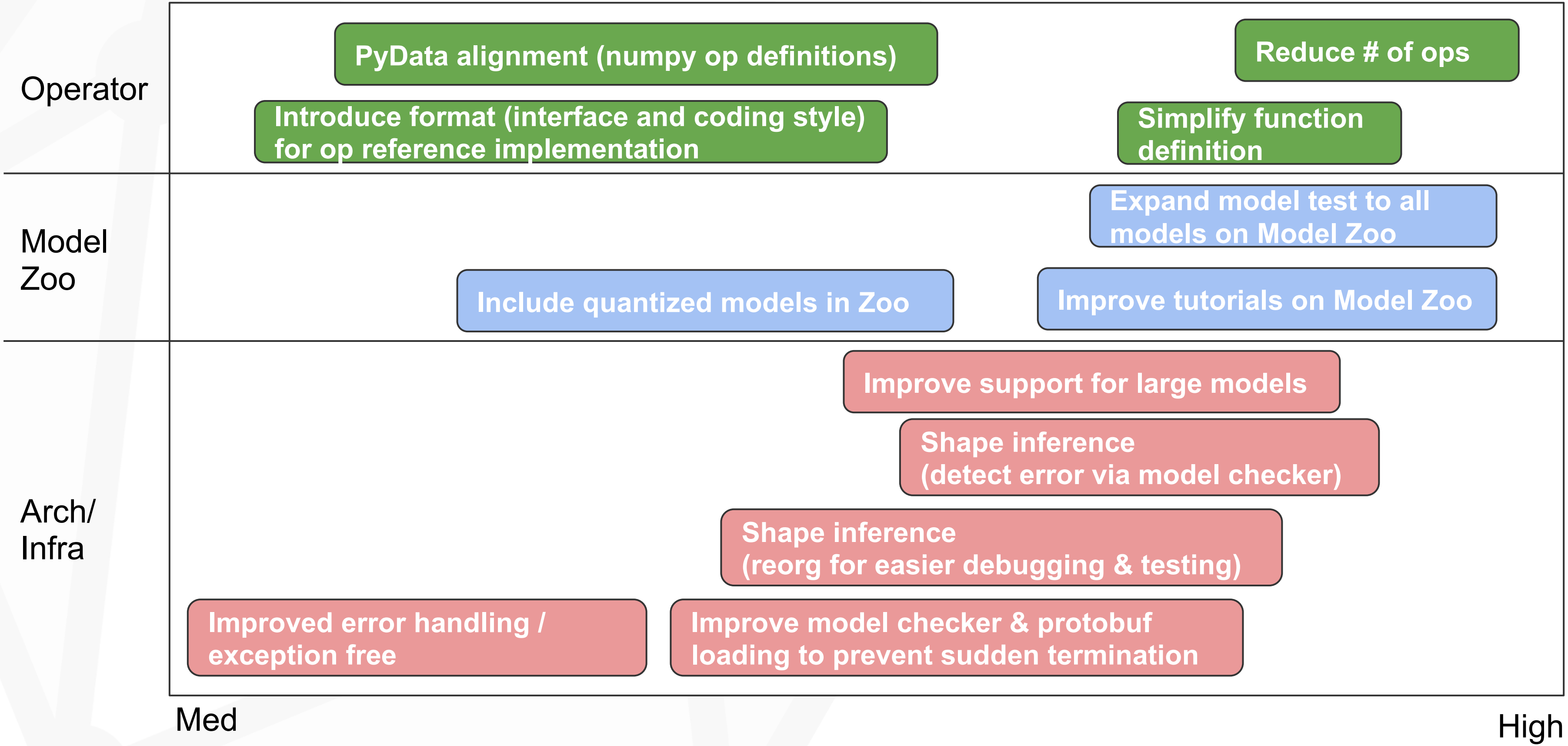
onnx.ai/impact

Impact analysis

Cost analysis

6 weekly community discussions

Suggested features & their rated impact



Operator

PyData alignment (numpy op definitions)

Reduce # of ops

Introduce format (interface and coding style) for op reference implementation

Simplify function definition

Model Zoo

Expand model test to all models on Model Zoo

Include quantized models in Zoo

Improve tutorials on Model Zoo

Arch/ Infra

Improve support for large models

Shape inference (detect error via model checker)

Shape inference (reorg for easier debugging & testing)

Improved error handling / exception free

Improve model checker & protobuf loading to prevent sudden termination

Med

High

Questions?

ONNX

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

ZETANE

23 Slides/9 minutes

Extract the maximum benefits of ONNX to shorten your development cycle time and reduce guesswork

Patrick St-Amant
Co-founder and CTO
Zetane Systems

The content of this presentation is not a contribution under the CLA

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

Important problem for industrial adoption

https://www.analyticsinsight.net/what-is-inside-the-black-box-of-artificial-intelligence

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

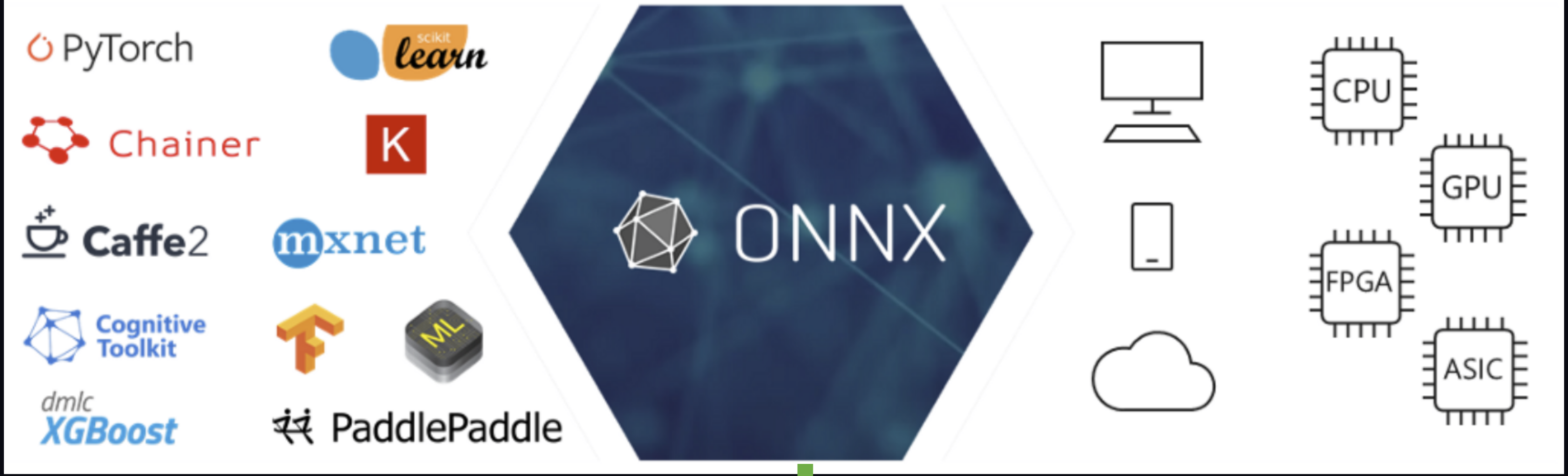
Lack of involvement in the ML process and no understanding of AI algorithms, code, and libraries resulting in lack of trust in what scientists are proposing.

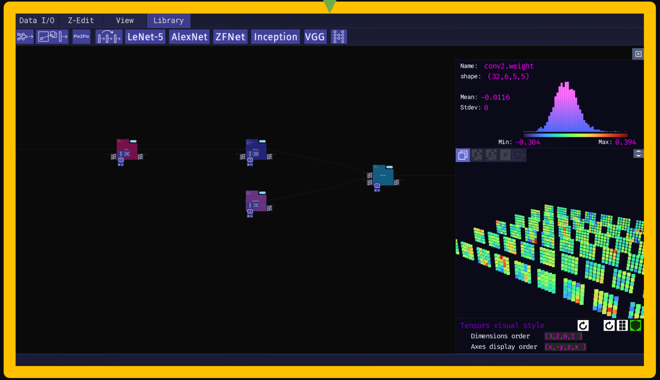
For business leaders

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

Data flows in ONNX model

Inputs





Name: conv2.weight
shape: (32,6,5,5)

Mean: -0.0116
Stdev: 0

Min: -0.304 Max: 0.394

Dimensions order (3,2,0,1)
Axes display order (x,-y,z,x)

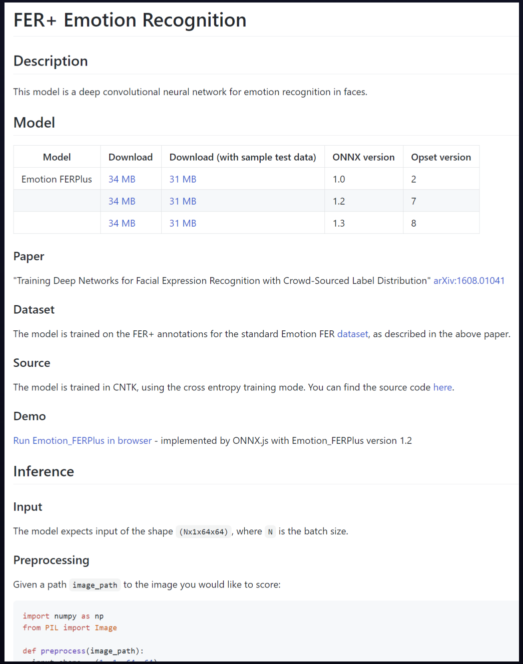
https://medium.com/codait/contribute-to-the-open-neural-network-exchange-onnx-5cfff6889761

Directly from the Python workflow

```
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

Model	Download	Download (with sample test data)	ONNX version	Opset version
Emotion FERPlus	34 MB	31 MB	1.0	2
	34 MB	31 MB	1.2	7
	34 MB	31 MB	1.3	8

Paper

"Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution" arXiv:1608.01041

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 in CNTK, using the cross entropy training mode. You can find the source code here.

Demo

Run Emotion_FERPlus in browser - implemented by ONNX.js with Emotion_FERPlus version 1.2

Inference

Input

The model expects input of the shape `(Nx1x64x64)`, where `N` is the batch size.

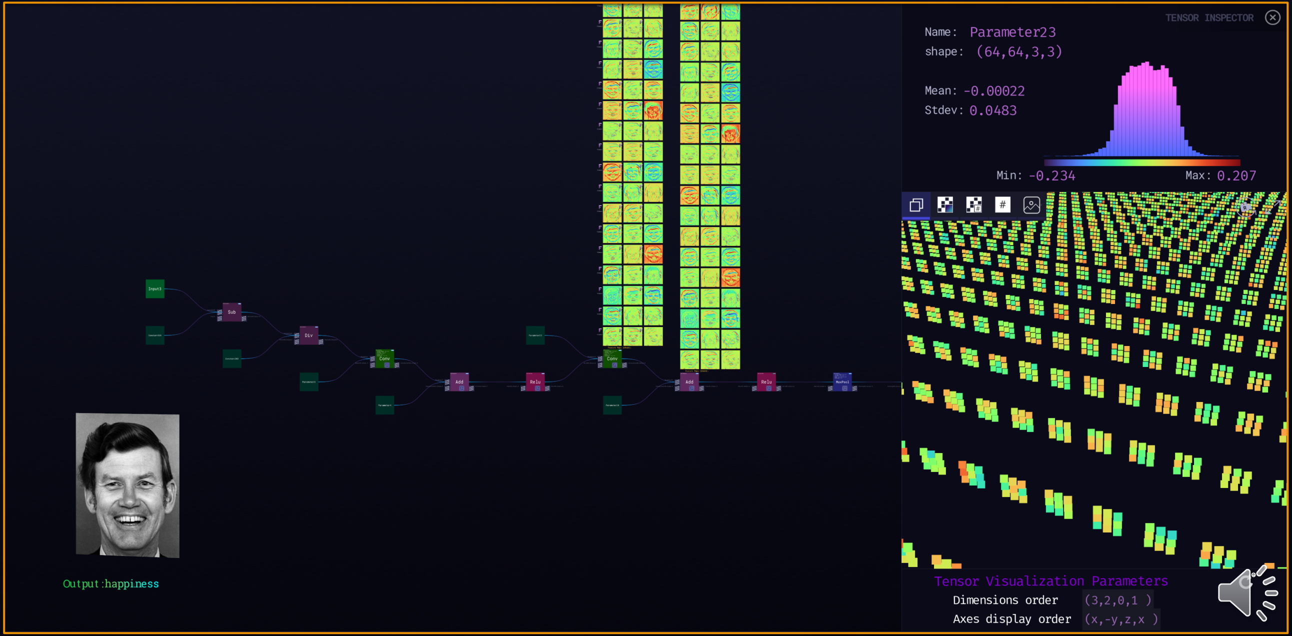
Preprocessing

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

```
import numpy as np
from PIL import Image

def preprocess(image_path):
```

https://github.com/onnx/models/tree/master/vision/body_analysis/emotion_ferplus

python™



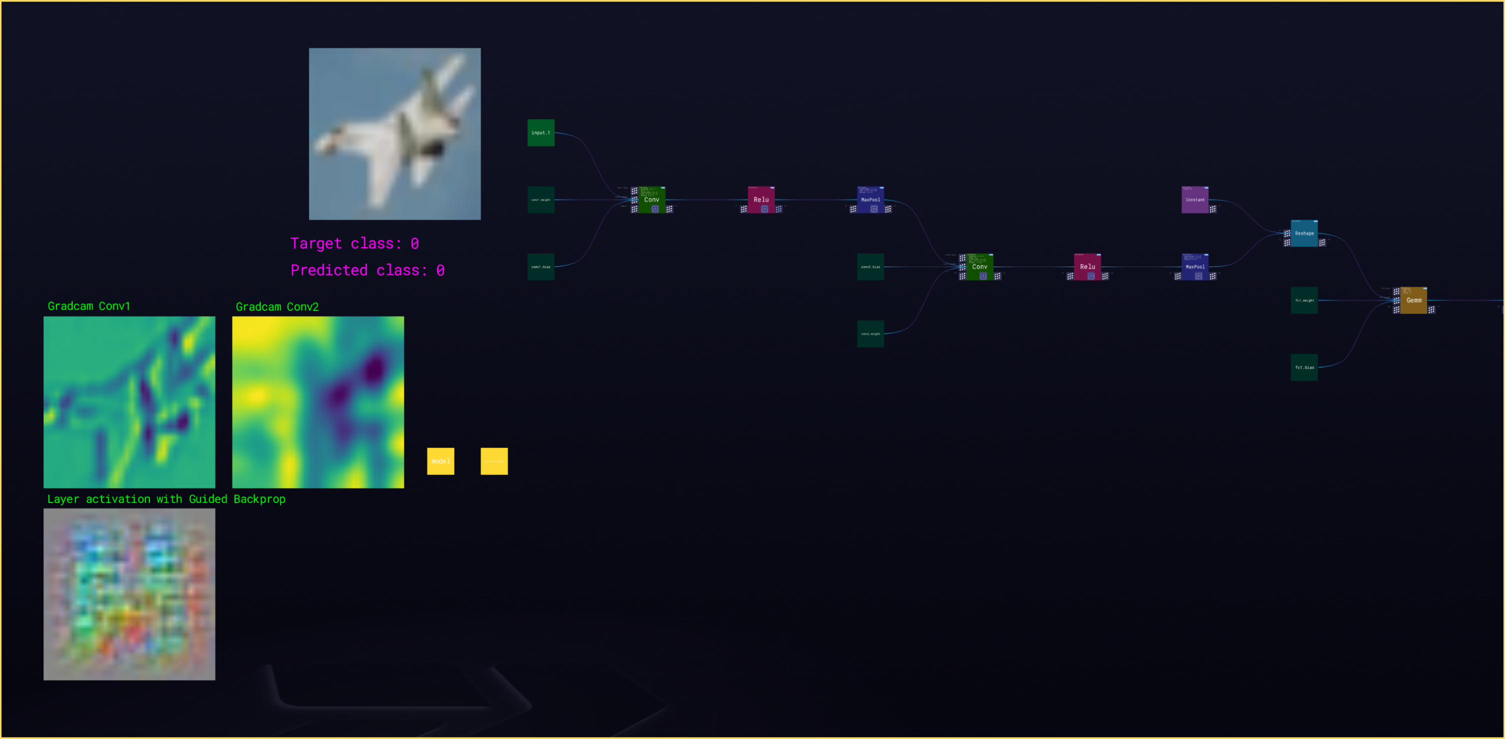
Output:happiness

```
Name: Parameter23
shape: (64,64,3,3)

Mean: -0.00022
Stdev: 0.0483

Min: -0.234          Max: 0.207
```

Tensor Visualization Parameters

```
Dimensions order    (3,2,0,1 )
Axes display order  (x,-y,z,x )
```

ONNX graph and API components



Target class: 0

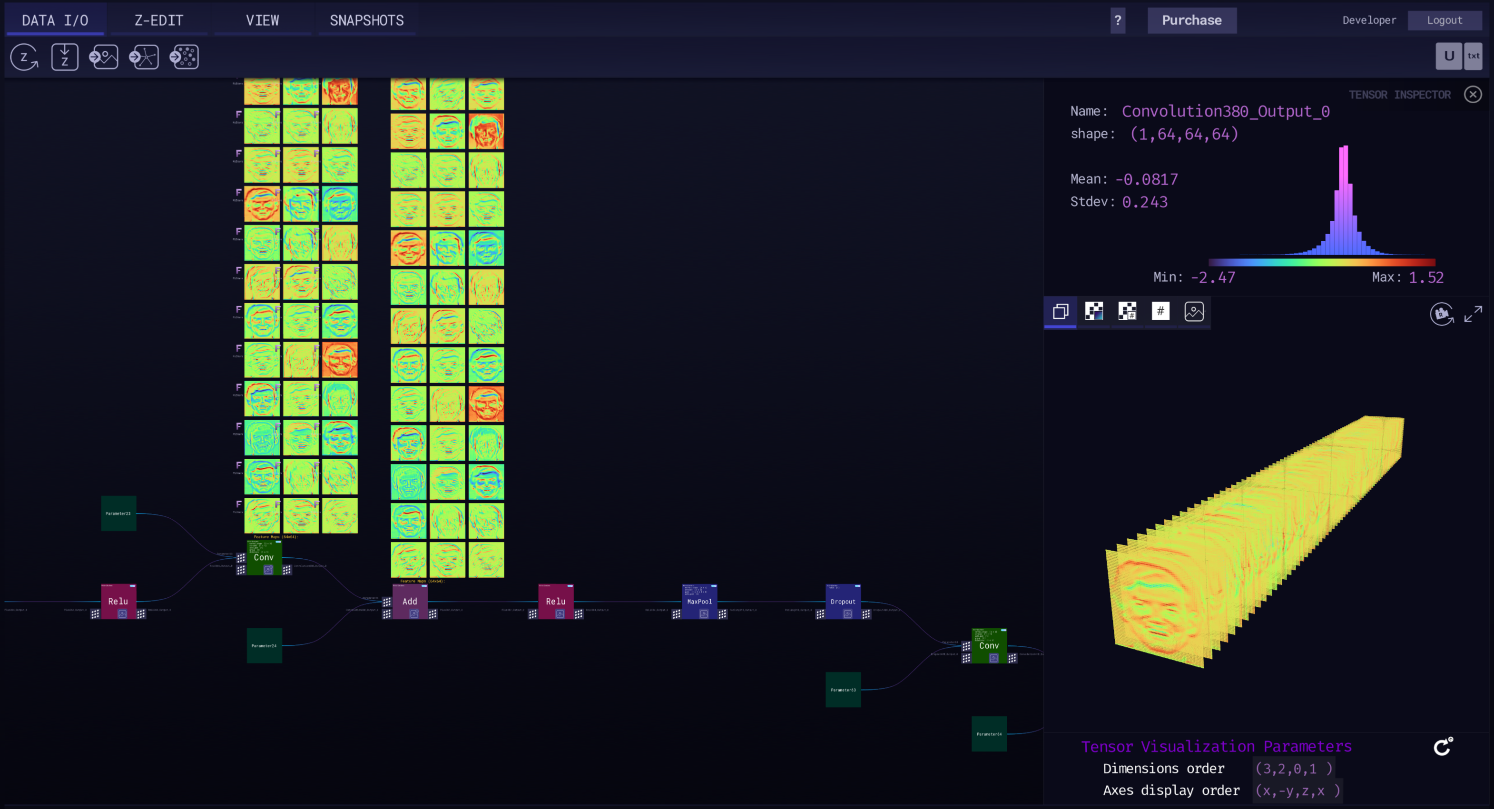
Predicted class: 0

Python-Zetane API components:

- Image
- Pointcloud
- 3D meshes
- Numpy
- Video
- Text
- Metric
- Chart
- Panel
- UI elements

Tensors

one-click access, visualizations and statistics



TENSOR INSPECTOR

Name: Convolution380_Output_0

shape: (1,64,64,64)

Mean: -0.0817

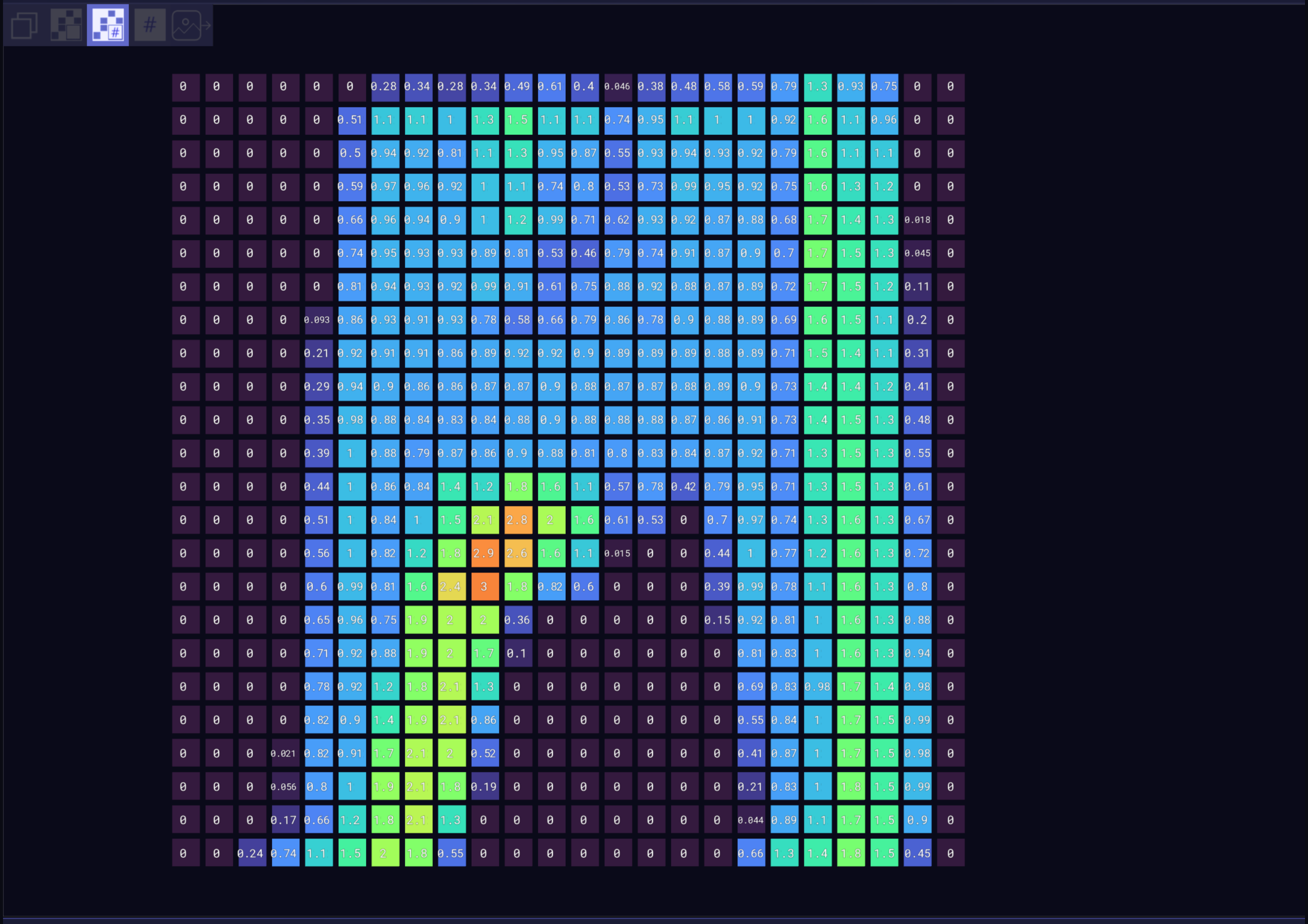
Stdev: 0.243

Min: -2.47 Max: 1.52

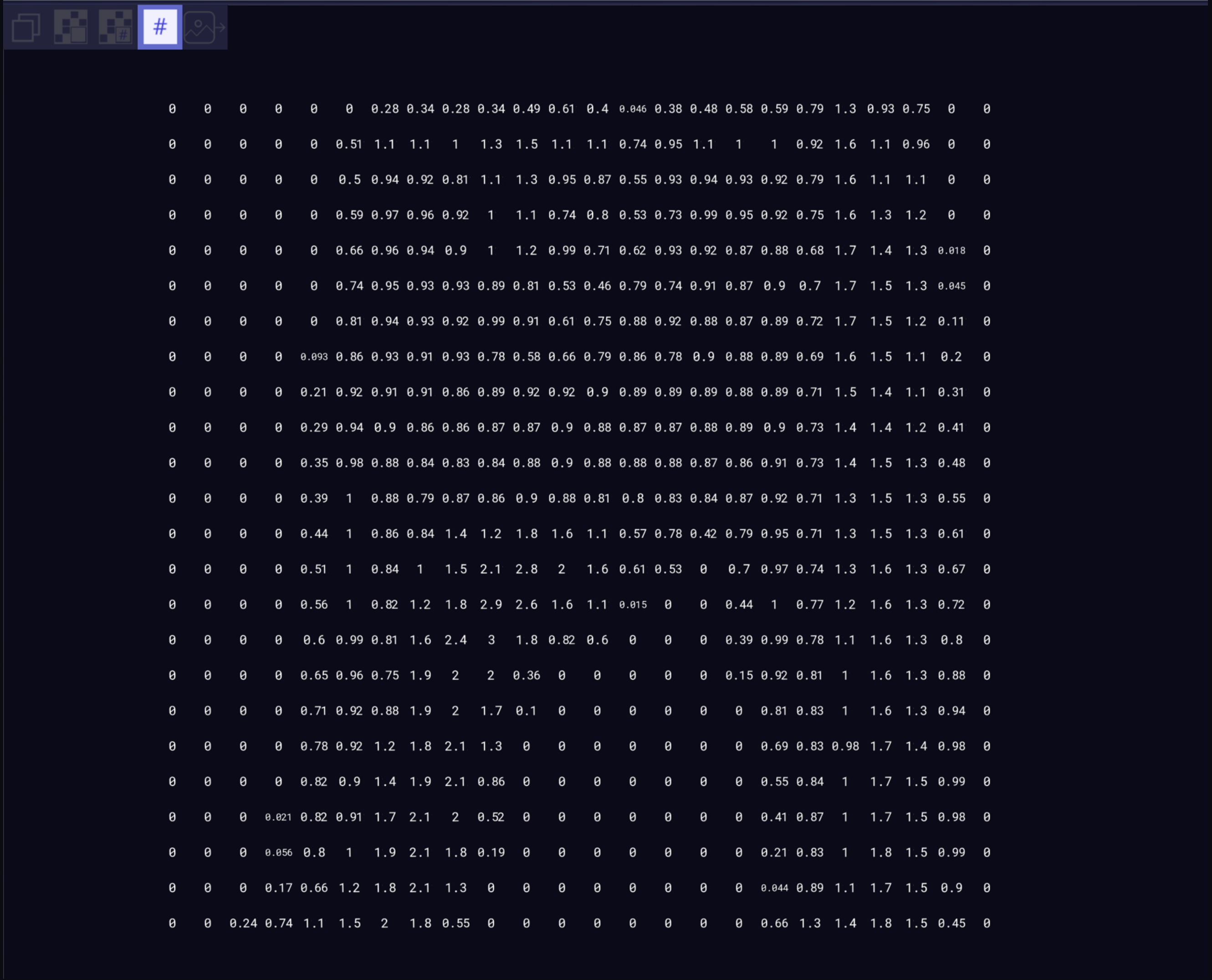
Tensor Visualization Parameters

Dimensions order (3,2,0,1)

Axes display order (x,-y,z,x)



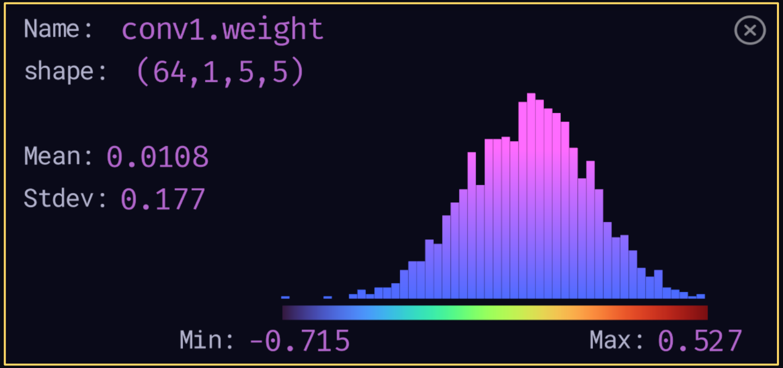
0	0	0	0	0	0	0.28	0.34	0.28	0.34	0.49	0.61	0.4	0.046	0.38	0.48	0.58	0.59	0.79	1.3	0.93	0.75	0	0
0	0	0	0	0	0.51	1.1	1.1	1	1.3	1.5	1.1	1.1	0.74	0.95	1.1	1	1	0.92	1.6	1.1	0.96	0	0
0	0	0	0	0	0.5	0.94	0.92	0.81	1.1	1.3	0.95	0.87	0.55	0.93	0.94	0.93	0.92	0.79	1.6	1.1	1.1	0	0
0	0	0	0	0	0.59	0.97	0.96	0.92	1	1.1	0.74	0.8	0.53	0.73	0.99	0.95	0.92	0.75	1.6	1.3	1.2	0	0
0	0	0	0	0	0.66	0.96	0.94	0.9	1	1.2	0.99	0.71	0.62	0.93	0.92	0.87	0.88	0.68	1.7	1.4	1.3	0.018	0
0	0	0	0	0	0.74	0.95	0.93	0.93	0.89	0.81	0.53	0.46	0.79	0.74	0.91	0.87	0.9	0.7	1.7	1.5	1.3	0.045	0
0	0	0	0	0	0.81	0.94	0.93	0.92	0.99	0.91	0.61	0.75	0.88	0.92	0.88	0.87	0.89	0.72	1.7	1.5	1.2	0.11	0
0	0	0	0	0.093	0.86	0.93	0.91	0.93	0.78	0.58	0.66	0.79	0.86	0.78	0.9	0.88	0.89	0.69	1.6	1.5	1.1	0.2	0
0	0	0	0	0.21	0.92	0.91	0.91	0.86	0.89	0.92	0.92	0.9	0.89	0.89	0.89	0.88	0.89	0.71	1.5	1.4	1.1	0.31	0
0	0	0	0	0.29	0.94	0.9	0.86	0.86	0.87	0.87	0.9	0.88	0.87	0.87	0.88	0.89	0.9	0.73	1.4	1.4	1.2	0.41	0
0	0	0	0	0.35	0.98	0.88	0.84	0.83	0.84	0.88	0.9	0.88	0.88	0.88	0.87	0.86	0.91	0.73	1.4	1.5	1.3	0.48	0
0	0	0	0	0.39	1	0.88	0.79	0.87	0.86	0.9	0.88	0.81	0.8	0.83	0.84	0.87	0.92	0.71	1.3	1.5	1.3	0.55	0
0	0	0	0	0.44	1	0.86	0.84	1.4	1.2	1.8	1.6	1.1	0.57	0.78	0.42	0.79	0.95	0.71	1.3	1.5	1.3	0.61	0
0	0	0	0	0.51	1	0.84	1	1.5	2.1	2.8	2	1.6	0.61	0.53	0	0.7	0.97	0.74	1.3	1.6	1.3	0.67	0
0	0	0	0	0.56	1	0.82	1.2	1.8	2.9	2.6	1.6	1.1	0.015	0	0	0.44	1	0.77	1.2	1.6	1.3	0.72	0
0	0	0	0	0.6	0.99	0.81	1.6	2.4	3	1.8	0.82	0.6	0	0	0	0.39	0.99	0.78	1.1	1.6	1.3	0.8	0
0	0	0	0	0.65	0.96	0.75	1.9	2	2	0.36	0	0	0	0	0	0.15	0.92	0.81	1	1.6	1.3	0.88	0
0	0	0	0	0.71	0.92	0.88	1.9	2	1.7	0.1	0	0	0	0	0	0	0.81	0.83	1	1.6	1.3	0.94	0
0	0	0	0	0.78	0.92	1.2	1.8	2.1	1.3	0	0	0	0	0	0	0	0.69	0.83	0.98	1.7	1.4	0.98	0
0	0	0	0	0.82	0.9	1.4	1.9	2.1	0.86	0	0	0	0	0	0	0	0.55	0.84	1	1.7	1.5	0.99	0
0	0	0	0.021	0.82	0.91	1.7	2.1	2	0.52	0	0	0	0	0	0	0	0.41	0.87	1	1.7	1.5	0.98	0
0	0	0	0.056	0.8	1	1.9	2.1	1.8	0.19	0	0	0	0	0	0	0	0.21	0.83	1	1.8	1.5	0.99	0
0	0	0	0.17	0.66	1.2	1.8	2.1	1.3	0	0	0	0	0	0	0	0	0.044	0.89	1.1	1.7	1.5	0.9	0
0	0	0.24	0.74	1.1	1.5	2	1.8	0.55	0	0	0	0	0	0	0	0	0.66	1.3	1.4	1.8	1.5	0.45	0



0	0	0	0	0	0	0.28	0.34	0.28	0.34	0.49	0.61	0.4	0.046	0.38	0.48	0.58	0.59	0.79	1.3	0.93	0.75	0	0
0	0	0	0	0	0.51	1.1	1.1	1	1.3	1.5	1.1	1.1	0.74	0.95	1.1	1	1	0.92	1.6	1.1	0.96	0	0
0	0	0	0	0	0.5	0.94	0.92	0.81	1.1	1.3	0.95	0.87	0.55	0.93	0.94	0.93	0.92	0.79	1.6	1.1	1.1	0	0
0	0	0	0	0	0.59	0.97	0.96	0.92	1	1.1	0.74	0.8	0.53	0.73	0.99	0.95	0.92	0.75	1.6	1.3	1.2	0	0
0	0	0	0	0	0.66	0.96	0.94	0.9	1	1.2	0.99	0.71	0.62	0.93	0.92	0.87	0.88	0.68	1.7	1.4	1.3	0.018	0
0	0	0	0	0	0.74	0.95	0.93	0.93	0.89	0.81	0.53	0.46	0.79	0.74	0.91	0.87	0.9	0.7	1.7	1.5	1.3	0.045	0
0	0	0	0	0	0.81	0.94	0.93	0.92	0.99	0.91	0.61	0.75	0.88	0.92	0.88	0.87	0.89	0.72	1.7	1.5	1.2	0.11	0
0	0	0	0	0.093	0.86	0.93	0.91	0.93	0.78	0.58	0.66	0.79	0.86	0.78	0.9	0.88	0.89	0.69	1.6	1.5	1.1	0.2	0
0	0	0	0	0.21	0.92	0.91	0.91	0.86	0.89	0.92	0.92	0.9	0.89	0.89	0.89	0.88	0.89	0.71	1.5	1.4	1.1	0.31	0
0	0	0	0	0.29	0.94	0.9	0.86	0.86	0.87	0.87	0.9	0.88	0.87	0.87	0.88	0.89	0.9	0.73	1.4	1.4	1.2	0.41	0
0	0	0	0	0.35	0.98	0.88	0.84	0.83	0.84	0.88	0.9	0.88	0.88	0.88	0.87	0.86	0.91	0.73	1.4	1.5	1.3	0.48	0
0	0	0	0	0.39	1	0.88	0.79	0.87	0.86	0.9	0.88	0.81	0.8	0.83	0.84	0.87	0.92	0.71	1.3	1.5	1.3	0.55	0
0	0	0	0	0.44	1	0.86	0.84	1.4	1.2	1.8	1.6	1.1	0.57	0.78	0.42	0.79	0.95	0.71	1.3	1.5	1.3	0.61	0
0	0	0	0	0.51	1	0.84	1	1.5	2.1	2.8	2	1.6	0.61	0.53	0	0.7	0.97	0.74	1.3	1.6	1.3	0.67	0
0	0	0	0	0.56	1	0.82	1.2	1.8	2.9	2.6	1.6	1.1	0.015	0	0	0.44	1	0.77	1.2	1.6	1.3	0.72	0
0	0	0	0	0.6	0.99	0.81	1.6	2.4	3	1.8	0.82	0.6	0	0	0	0.39	0.99	0.78	1.1	1.6	1.3	0.8	0
0	0	0	0	0.65	0.96	0.75	1.9	2	2	0.36	0	0	0	0	0	0.15	0.92	0.81	1	1.6	1.3	0.88	0
0	0	0	0	0.71	0.92	0.88	1.9	2	1.7	0.1	0	0	0	0	0	0	0.81	0.83	1	1.6	1.3	0.94	0
0	0	0	0	0.78	0.92	1.2	1.8	2.1	1.3	0	0	0	0	0	0	0	0.69	0.83	0.98	1.7	1.4	0.98	0
0	0	0	0	0.82	0.9	1.4	1.9	2.1	0.86	0	0	0	0	0	0	0	0.55	0.84	1	1.7	1.5	0.99	0
0	0	0	0.021	0.82	0.91	1.7	2.1	2	0.52	0	0	0	0	0	0	0	0.41	0.87	1	1.7	1.5	0.98	0
0	0	0	0.056	0.8	1	1.9	2.1	1.8	0.19	0	0	0	0	0	0	0	0.21	0.83	1	1.8	1.5	0.99	0
0	0	0	0.17	0.66	1.2	1.8	2.1	1.3	0	0	0	0	0	0	0	0	0.044	0.89	1.1	1.7	1.5	0.9	0
0	0	0.24	0.74	1.1	1.5	2	1.8	0.55	0	0	0	0	0	0	0	0	0.66	1.3	1.4	1.8	1.5	0.45	0

```
Attributes:
  dilations  [1 x 1]
  group  [1]
  kernel_shape  [5 x 5]
  pads  [2 x 2 x 2 x 2]
  strides  [1 x 1]
```

Conv



Name: conv1.weight

shape: (64,1,5,5)

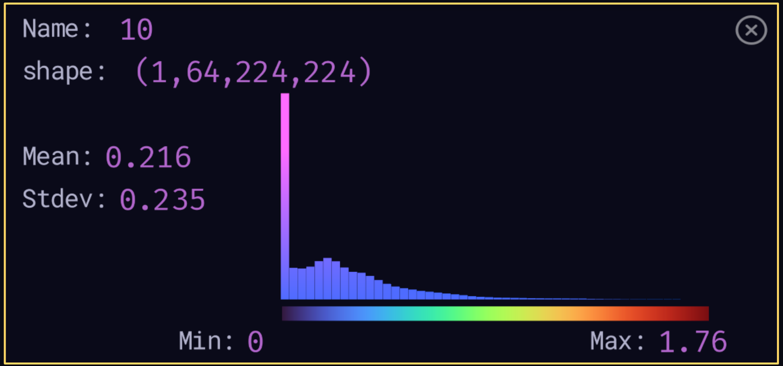
Mean: 0.0108

Stdev: 0.177

Min: -0.715

Max: 0.527

Relu



Name: 10

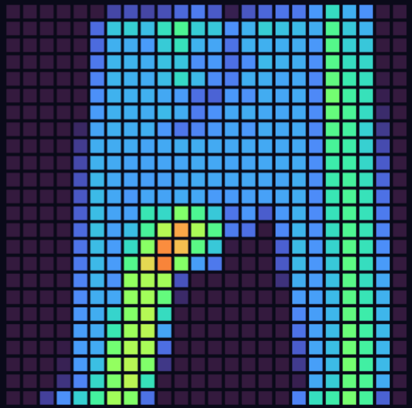
shape: (1,64,224,224)

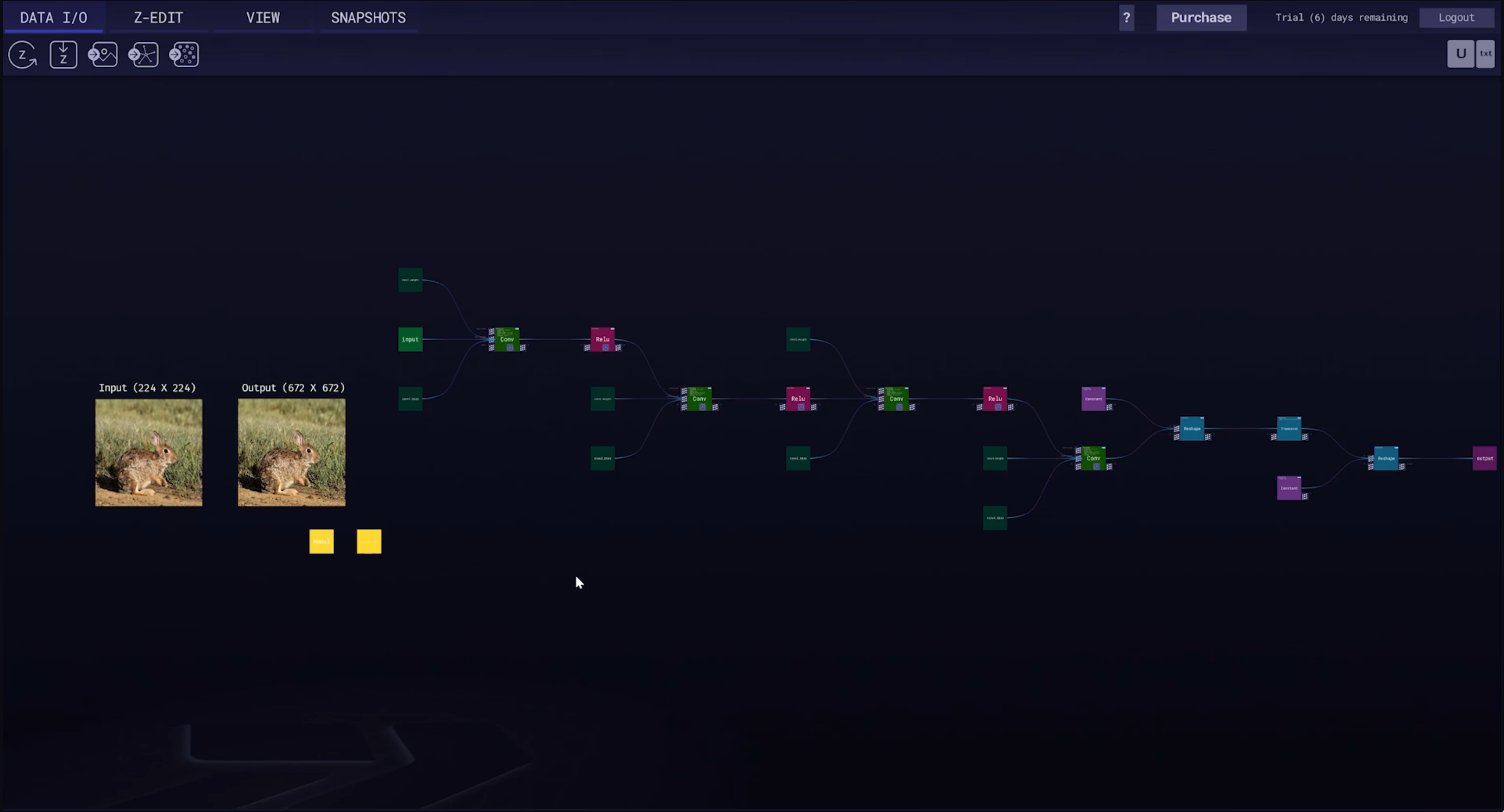
Mean: 0.216

Stdev: 0.235

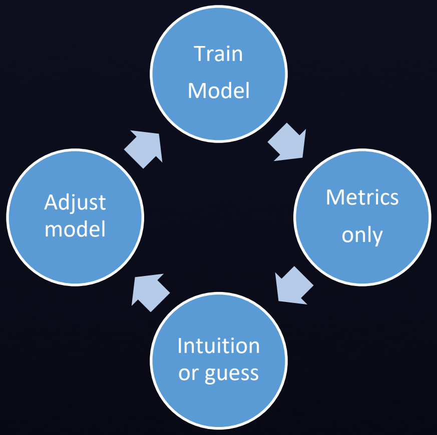
Min: 0

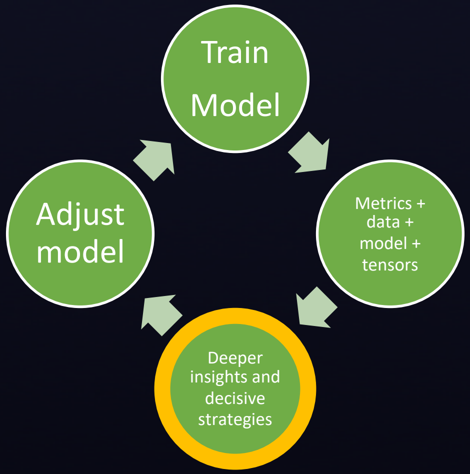
Max: 1.76




DATA I/O
Z-EDIT
VIEW
SNAPSHOTS
?
Purchase
Trial (6) days remaining
Logout
U
txt
Input (224 X 224)
Output (672 X 672)
conv1.weight
input
conv1.bias
Conv
Relu
conv2.weight
conv2.bias
Conv
Relu
Conv
Relu
Constant
Conv
Reshape
Transpose
Constant
Reshape
output

Reduce guesswork and shorten dev time




Train Model
Metrics + data + model + tensors
Deeper insights and decisive strategies
Adjust model

Focus on the interaction between the data and model

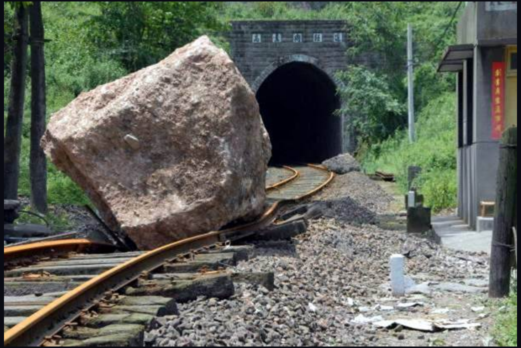
More decisive model improvements

Stop training early

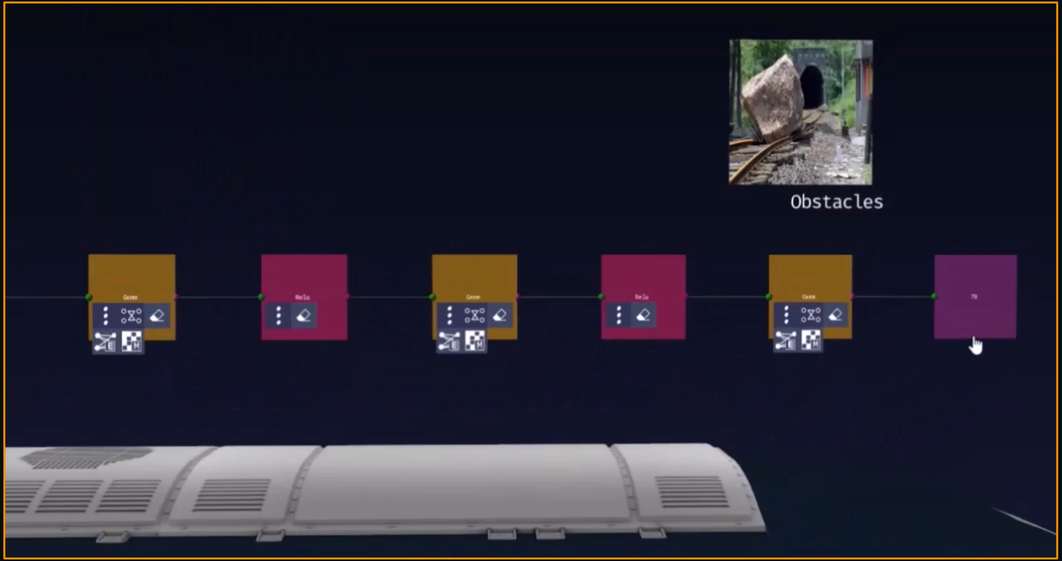
Less wasted time retraining

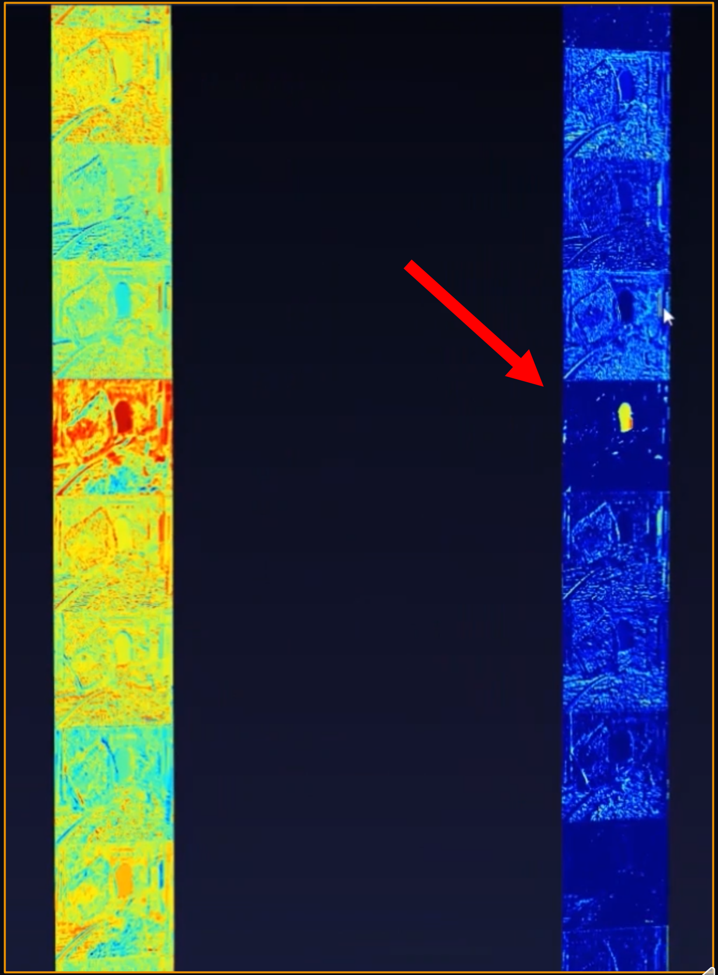
Increase trust and safety

Autonomous train use case



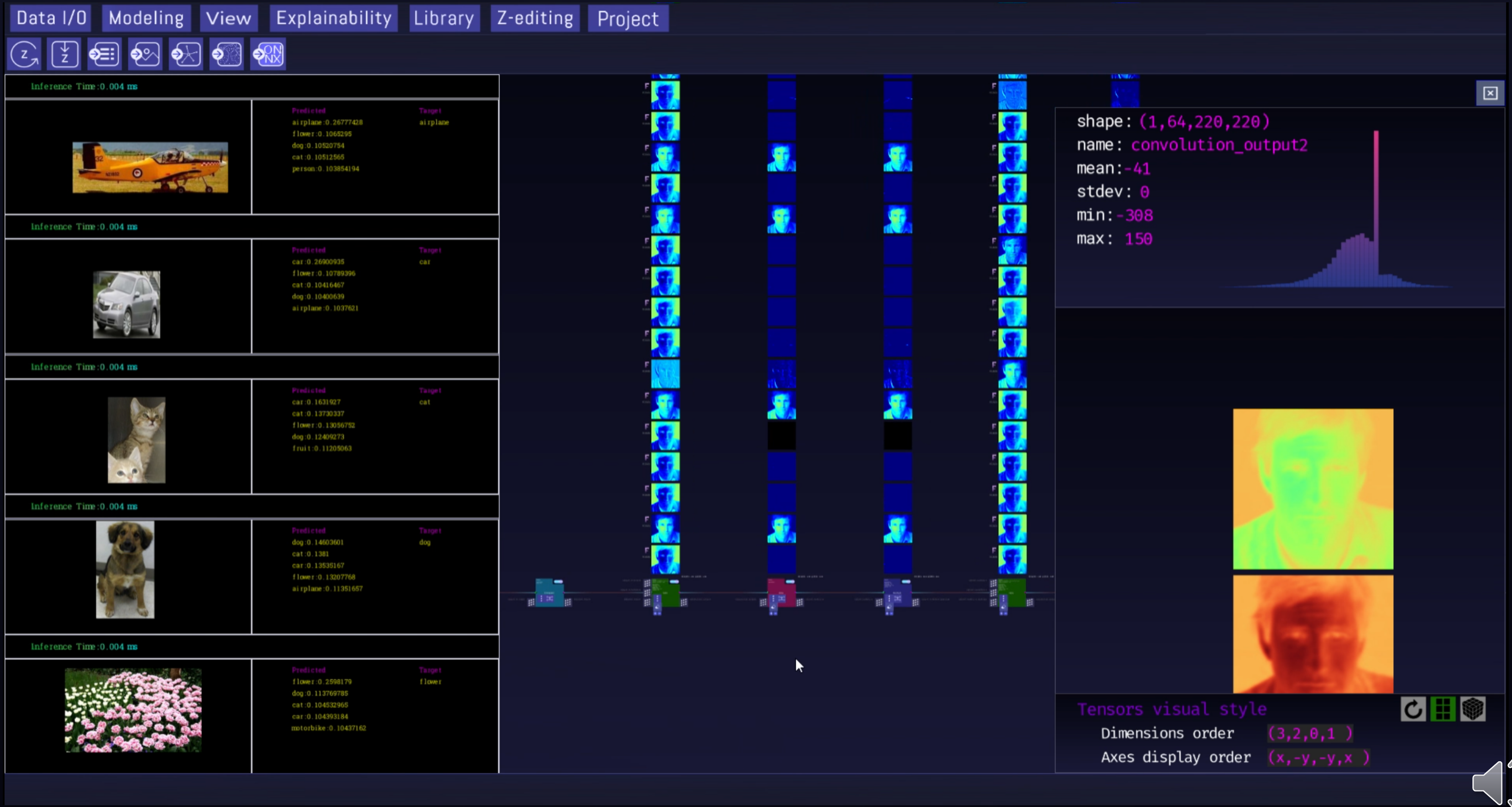
https://www.eeri.org/2008/05/wenchuan/02-12/





Inference dashboard

Focus on outliers

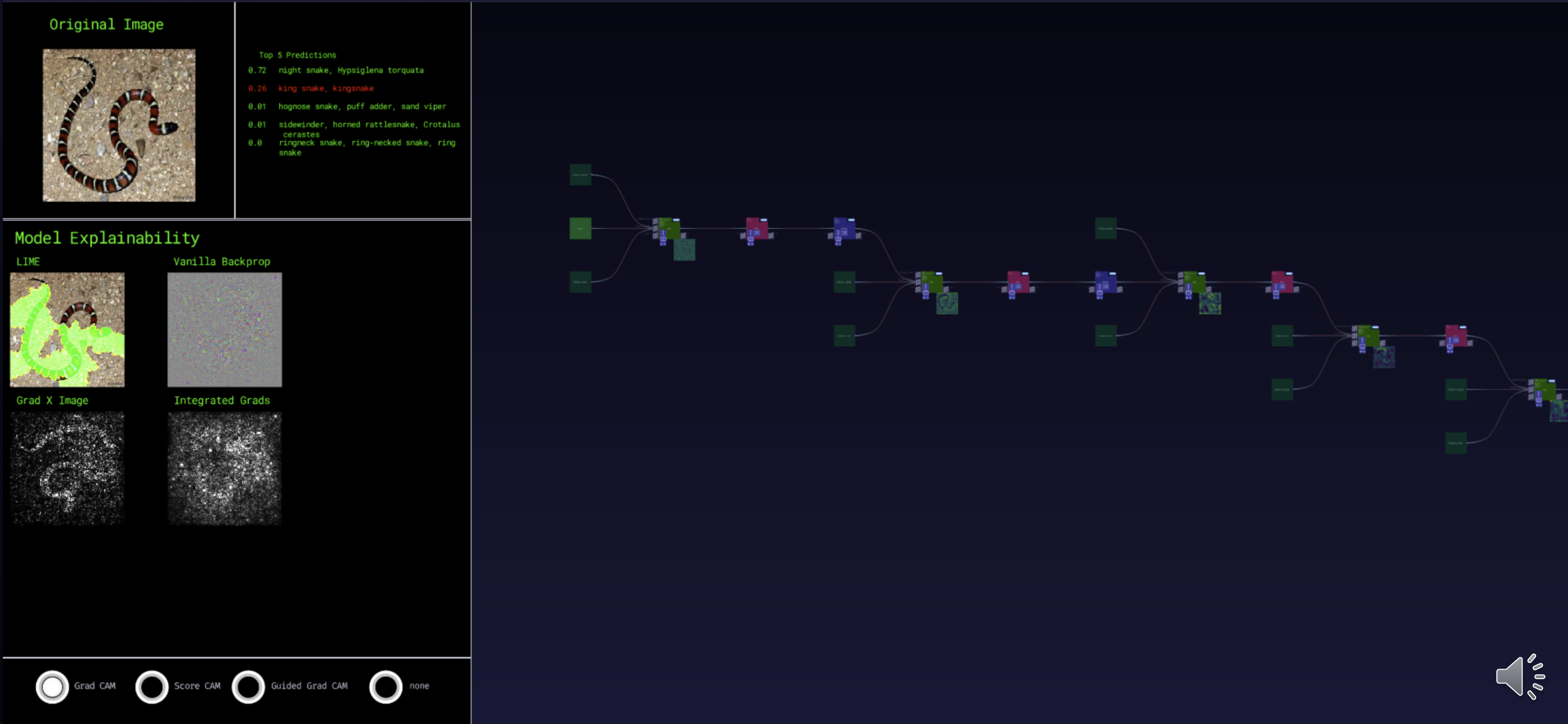


```
Inference Time:0.004 ms
Predicted                     Target
airplane:0.26777428           airplane
flower:0.1065295
dog:0.10520754
cat:0.10512565
person:0.103854194

Inference Time:0.004 ms
Predicted                     Target
car:0.26900935                car
flower:0.10789396
cat:0.10416467
dog:0.10400639
airplane:0.1037621

Inference Time:0.004 ms
Predicted                     Target
car:0.1631927                 cat
cat:0.13730337
flower:0.13056752
dog:0.12409273
fruit:0.11205063

Inference Time:0.004 ms
Predicted                     Target
dog:0.14603601                dog
cat:0.1381
car:0.13535167
flower:0.13207768
airplane:0.11351657

Inference Time:0.004 ms
Predicted                     Target
flower:0.2598179              flower
dog:0.113769785
cat:0.104532965
car:0.104393184
motorbike:0.10437162
```

```
shape: (1,64,220,220)
name: convolution_output2
mean:-41
stdev: 0
min:-308
max: 150
```

```
Tensors visual style
  Dimensions order    (3,2,0,1 )
  Axes display order  (x,-y,-y,x )
```

xAI dashboard



Original Image

Top 5 Predictions

0.72	night snake, Hypsiglena torquata
0.26	king snake, kingsnake
0.01	hognose snake, puff adder, sand viper
0.01	sidewinder, horned rattlesnake, Crotalus cerastes
0.0	ringneck snake, ring-necked snake, ring snake

Model Explainability

LIME

Vanilla Backprop

Grad X Image

Integrated Grads

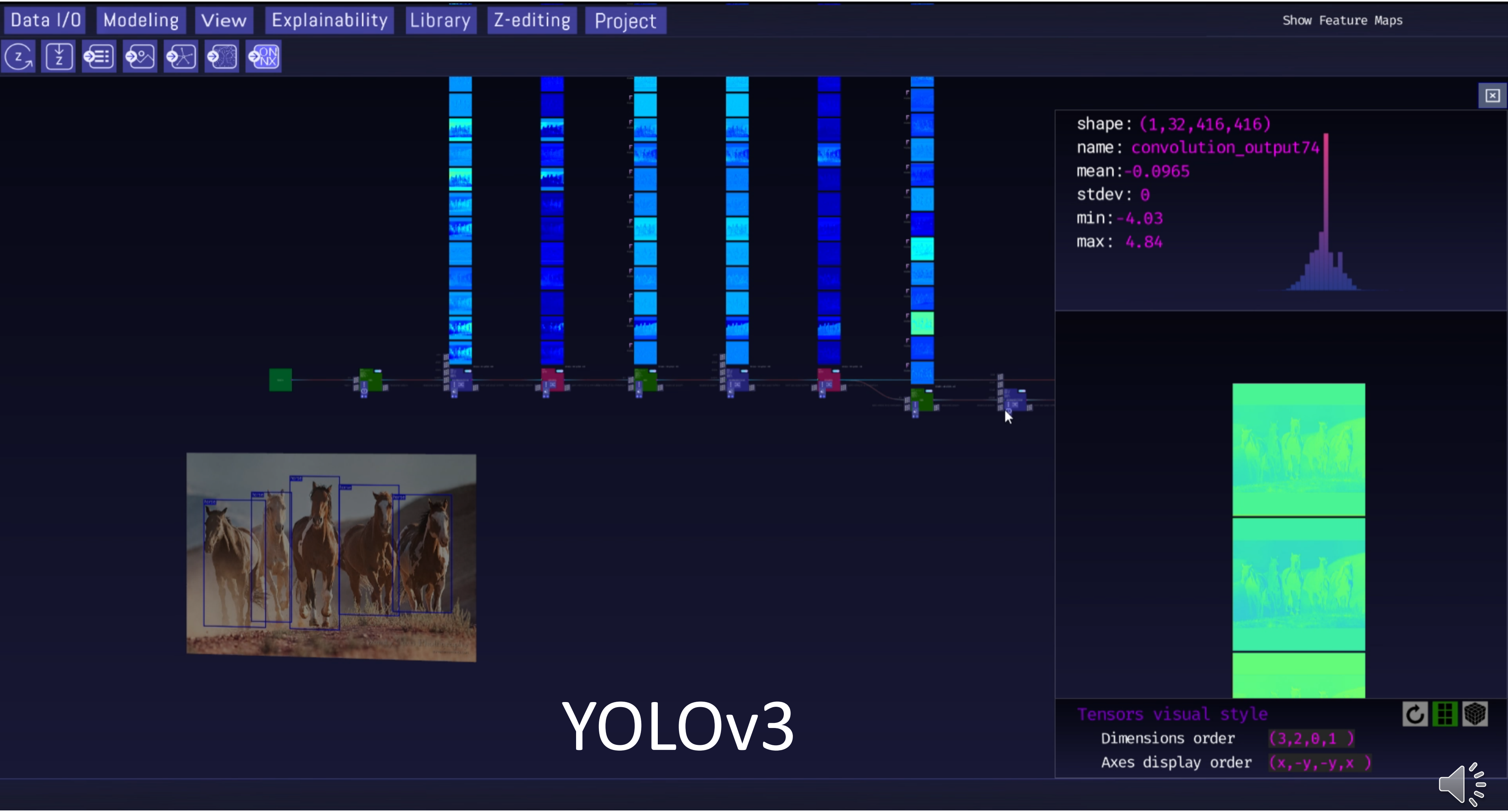
● Grad CAM ○ Score CAM ○ Guided Grad CAM ○ none

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



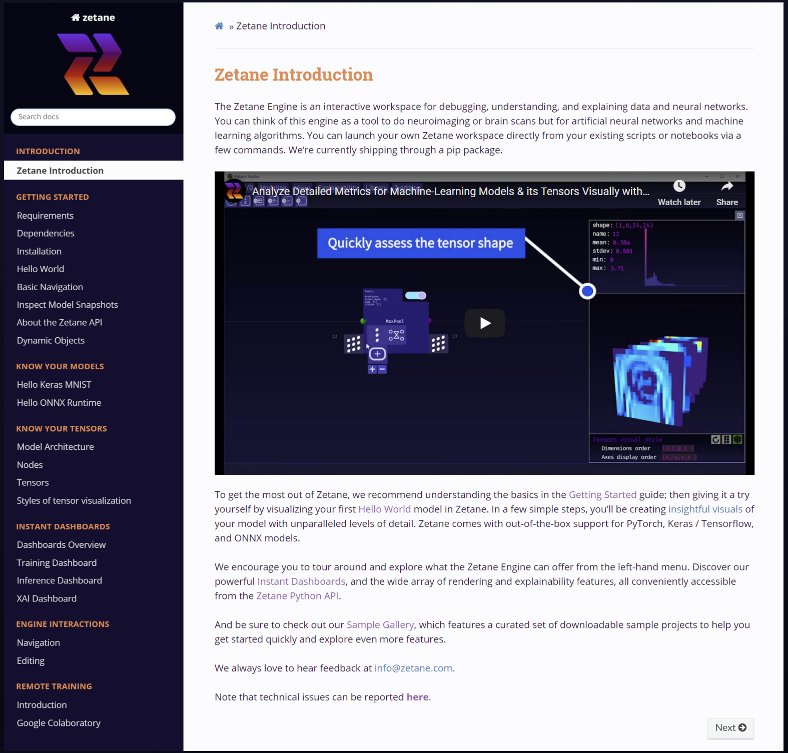
shape: (1,32,416,416)
name: convolution_output74
mean: -0.0965
stdev: 0
min: -4.03
max: 4.84

Tensors visual style
Dimensions order (3,2,0,1)
Axes display order (x,-y,-y,x)

YOLOv3

We recently launched the Zetane Engine

The community Zetane Viewer is coming soon



Zetane Introduction

The Zetane Engine is an interactive workspace for debugging, understanding, and explaining data and neural networks. You can think of this engine as a tool to do neuroimaging or brain scans but for artificial neural networks and machine learning algorithms. You can launch your own Zetane workspace directly from your existing scripts or notebooks via a few commands. We're currently shipping through a pip package.

To get the most out of Zetane, we recommend understanding the basics in the Getting Started guide; then giving it a try yourself by visualizing your first Hello World model in Zetane. In a few simple steps, you'll be creating insightful visuals of your model with unparalleled levels of detail. Zetane comes with out-of-the-box support for PyTorch, Keras / Tensorflow, and ONNX models.

We encourage you to tour around and explore what the Zetane Engine can offer from the left-hand menu. Discover our powerful Instant Dashboards, and the wide array of rendering and explainability features, all conveniently accessible from the Zetane Python API.

And be sure to check out our Sample Gallery, which features a curated set of downloadable sample projects to help you get started quickly and explore even more features.

We always love to hear feedback at info@zetane.com.

Note that technical issues can be reported here.

ZETANE

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

The content of this presentation is not a contribution under the

13 Slides/9 minutes

ONNX at OneFlow

Jianhao Zhang

OneFlow Inc.

About me

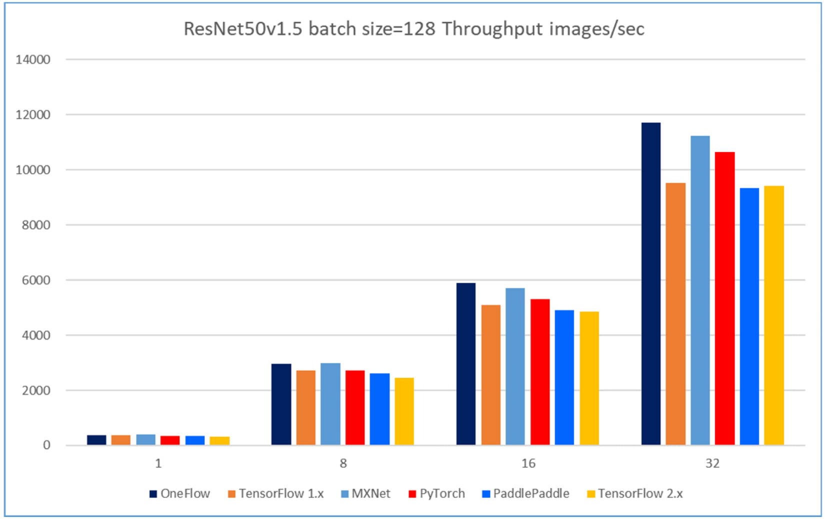
- GitHub: daquexian
- A developer at OneFlow Inc.
- An active contributor from ONNX community, a member of operator SIG
- ONNX-related work:
  - ONNX <-> OneFlow conversion (today's presentation)
  - onnx-simplifier
  - onnx/optimizer
  - resize and softmax op spec
- Also a presenter at ONNX workshop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

What is OneFlow

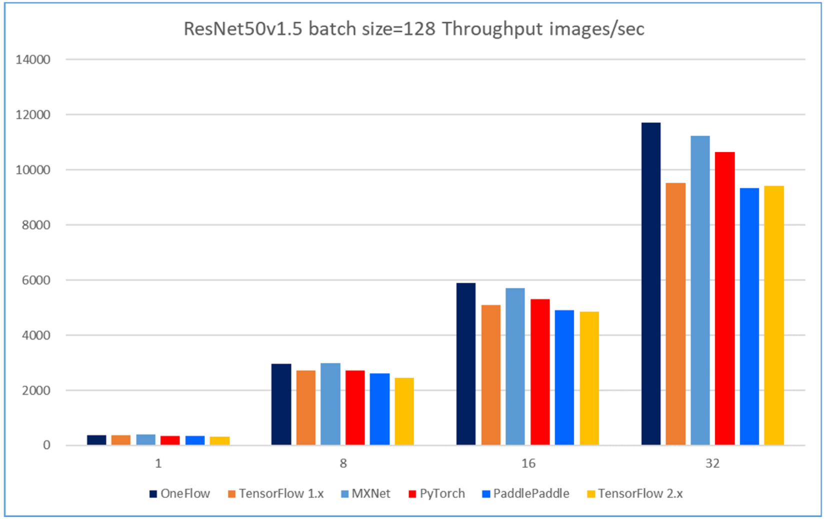
Oneflow is a brand-new open-source training framework focusing on distributed training. It makes distributed training on multi-machines and multi-devices as simple as on single device.

- 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

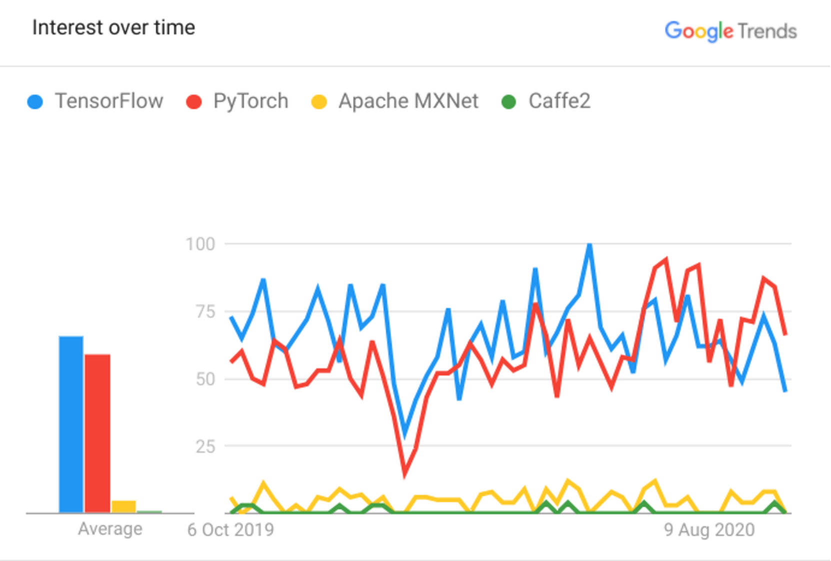


OneFlow Benchmark



The detailed benchmark report is public at https://github.com/Oneflow-Inc/DLPerf!

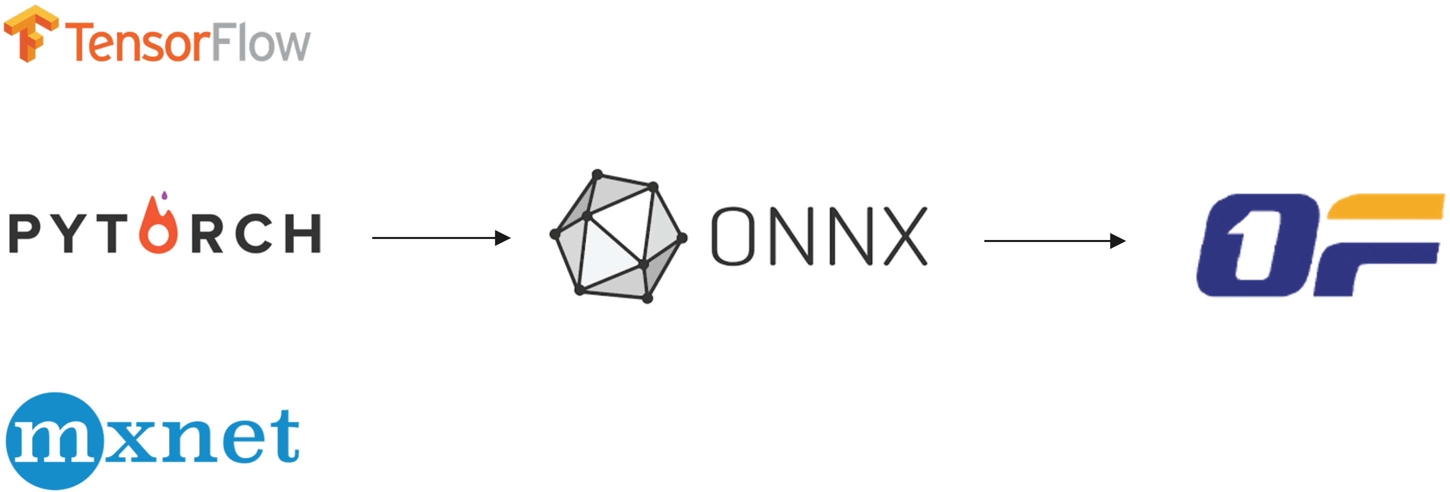
“Sounds Great, but..”



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

```
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)

```
# PyTorch code
import torchvision as tv
model = tv.models.resnet18()
```

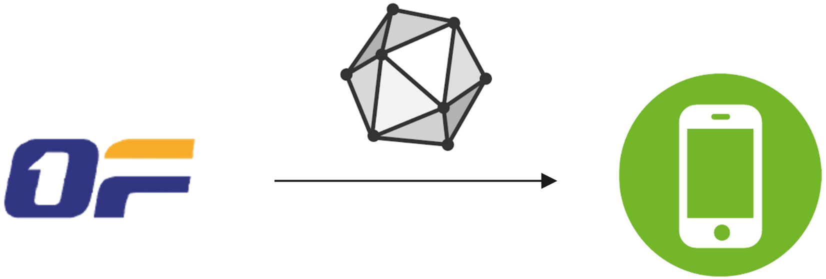
→

```
# 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)
.....
```

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

7 Slides/9 minutes

HUGGING FACE × ONNX RUNTIME

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

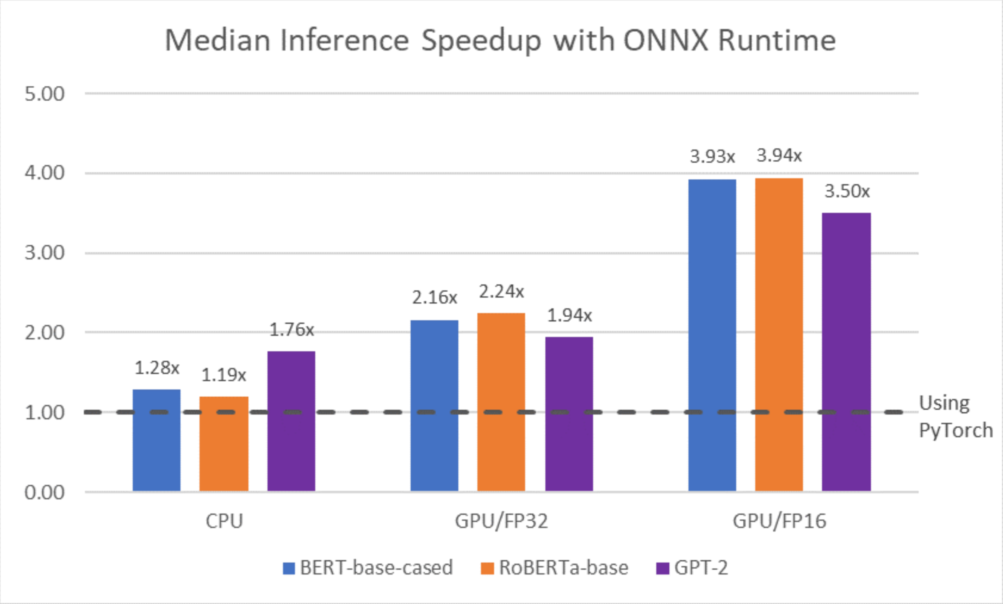
🤗 transformers	(+34,000★)
🤗 tokenizers	(+3,800 ★)
🤗 datasets	(+4,200 ★)

- 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

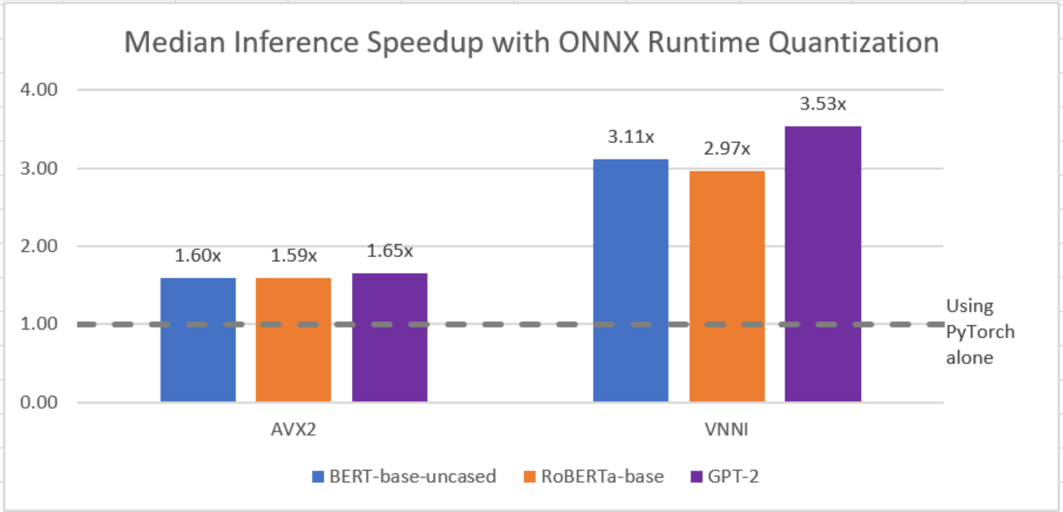
- Looking for a solution to export from PyTorch & TensorFlow.
- Initial integration with transformers to easily export wide variety of our models (25 architectures, from BERT to Reformer & more recently RAG).
- Leverage ONNX Runtime optimizations to speed-up inference on variety of hardwares and platformes.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

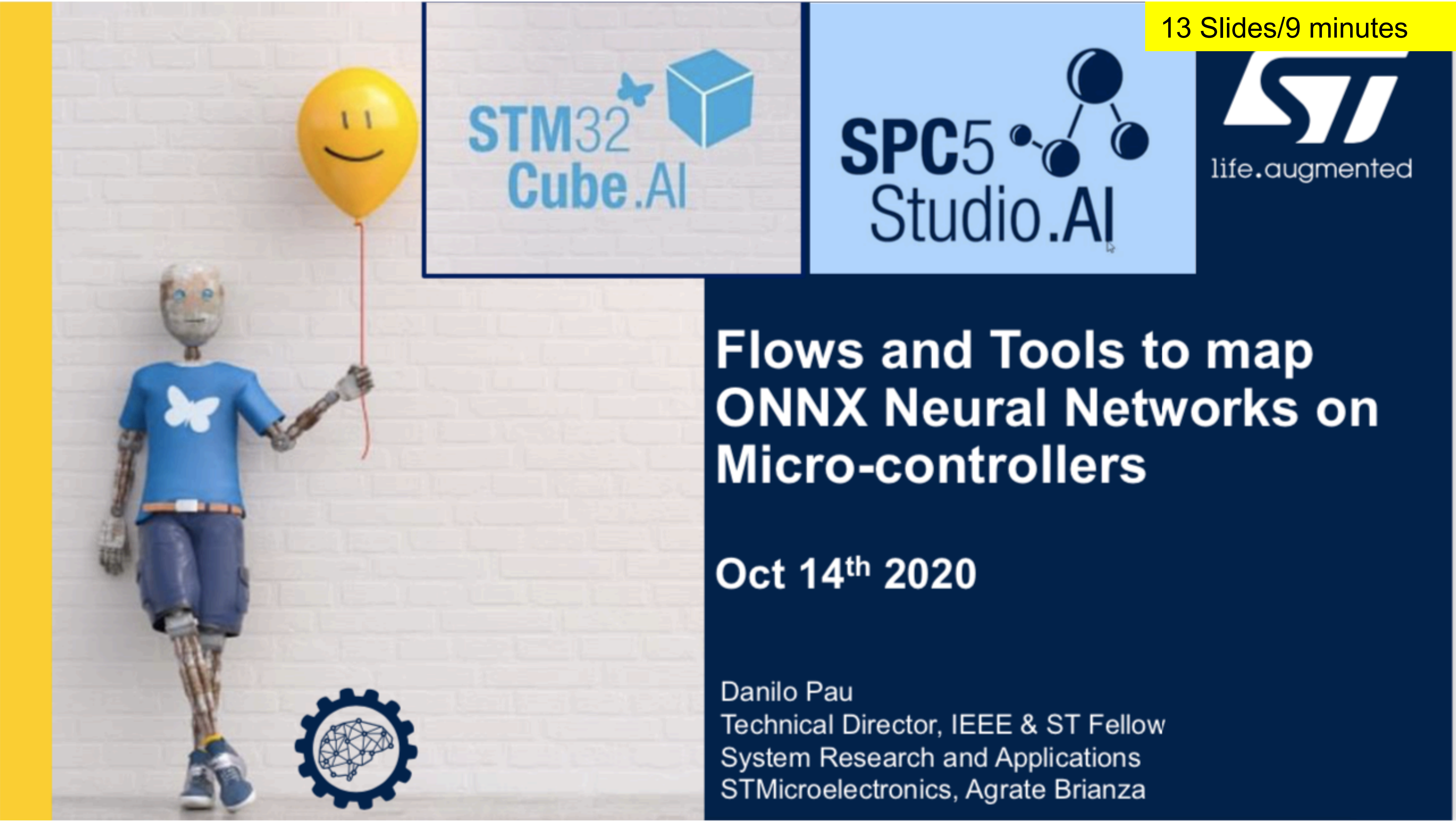
Conclusion

- 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.

13 Slides/9 minutes



STM32 Cube.AI

SPC5 Studio.AI

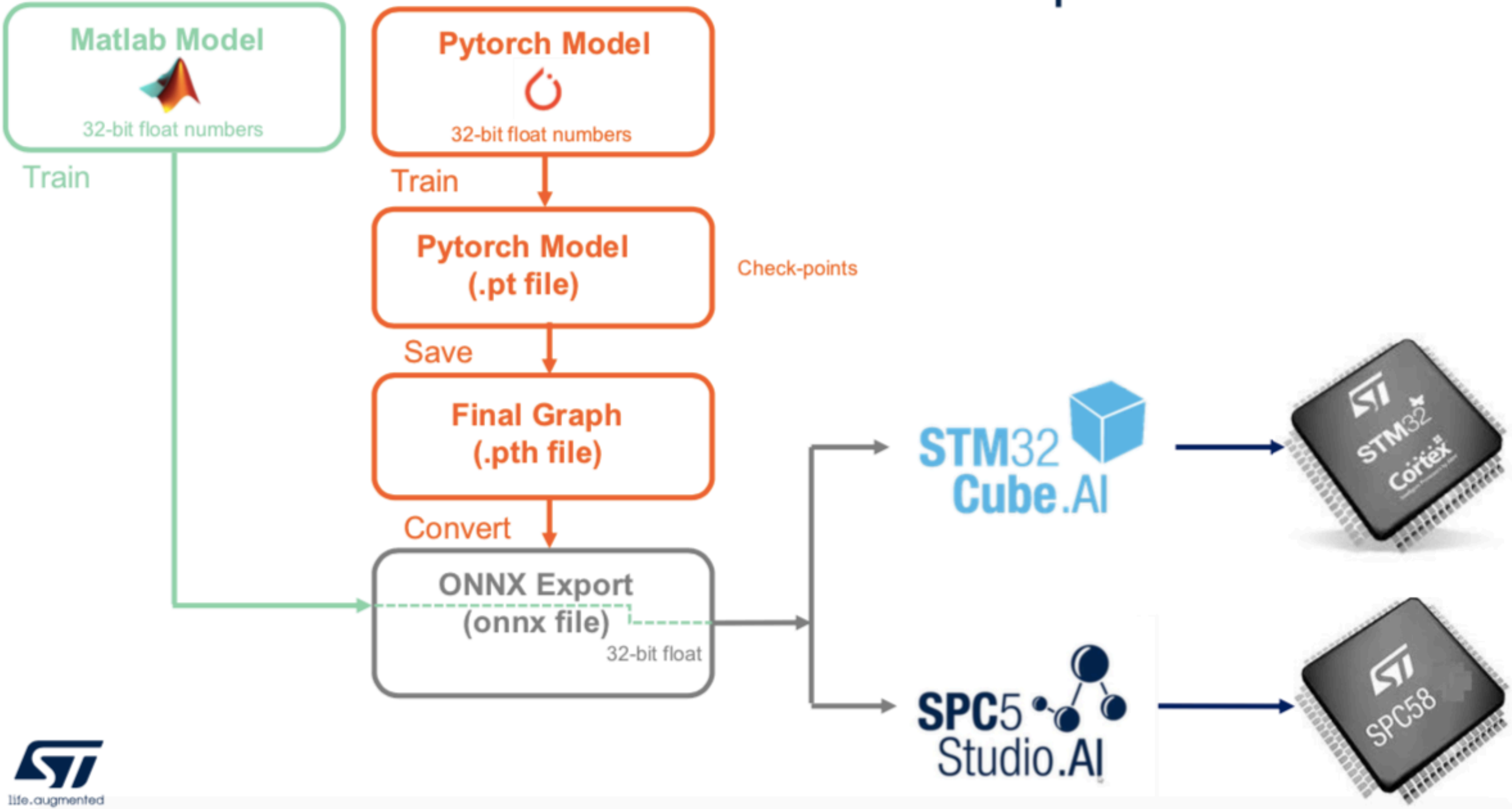
life.augmented

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

Development Flow in use



Matlab Model

32-bit float numbers

Train

Pytorch Model

32-bit float numbers

Train

Pytorch Model (.pt file)

Check-points

Save

Final Graph (.pth file)

Convert

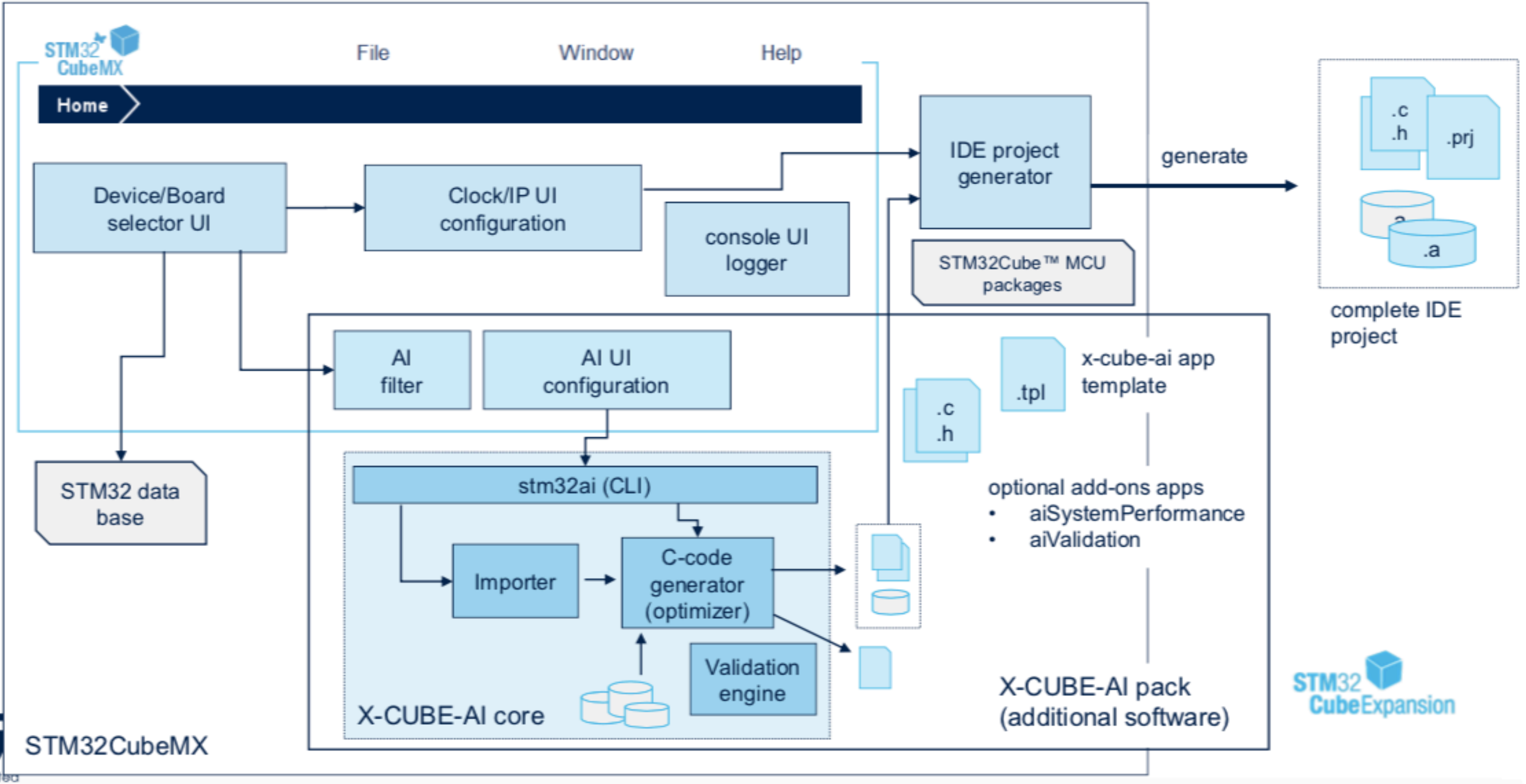
ONNX Export (onnx file)

32-bit float

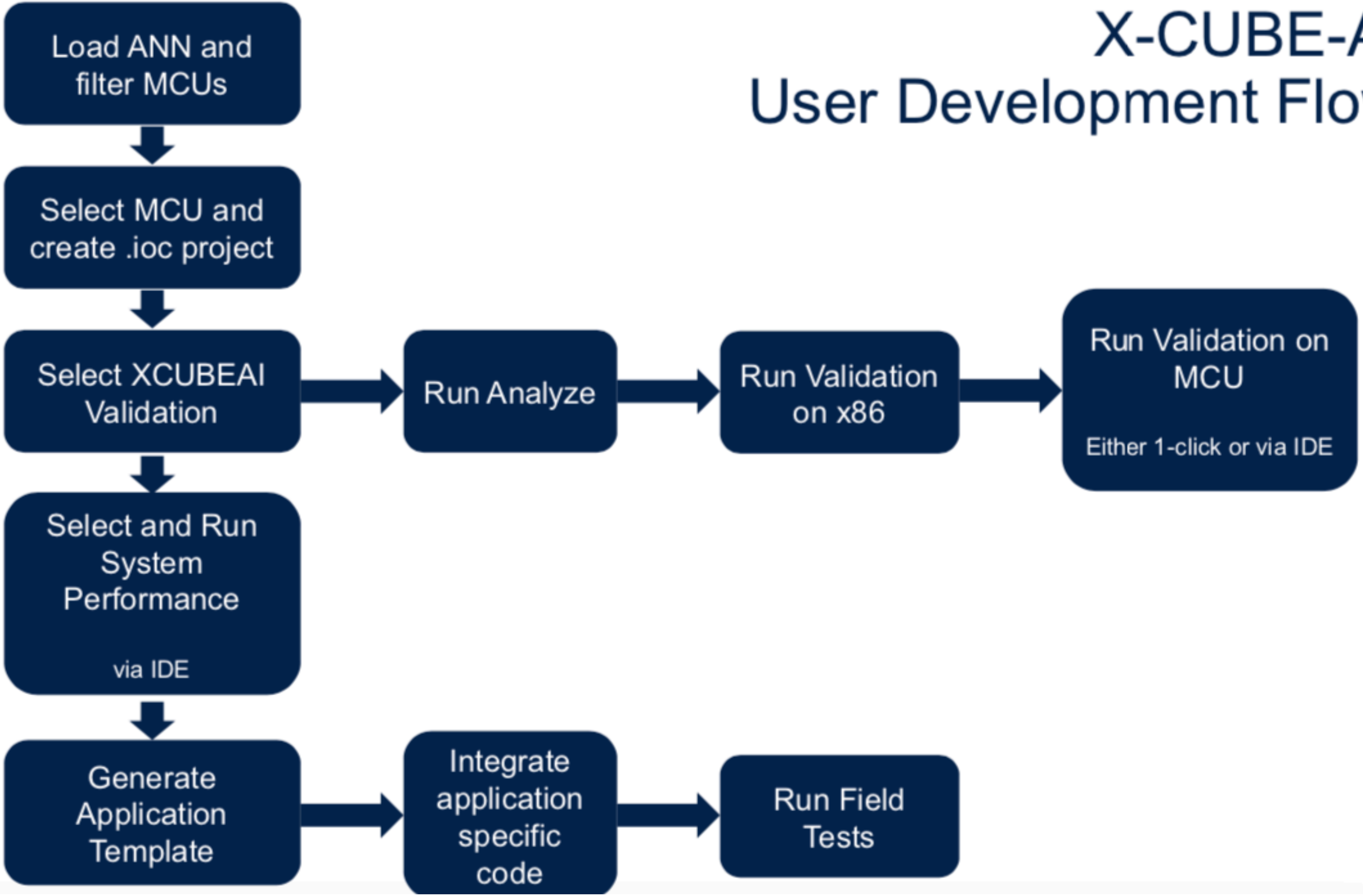
STM32 Cube.AI

SPC5 Studio.AI

X-CUBE-AI package as STM32CubeMX cube expansion



X-CUBE-AI User Development Flow



Load ANN and filter MCUs

Select MCU and create .ioc project

Select XCUBEAI Validation

Run Analyze

Run Validation on x86

Run Validation on MCU

Either 1-click or via IDE

Select and Run System Performance

via IDE

Generate Application Template

Integrate application specific code

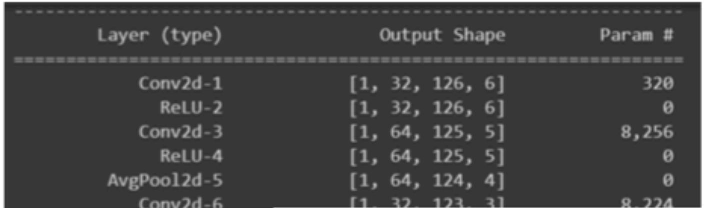
Run Field Tests

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

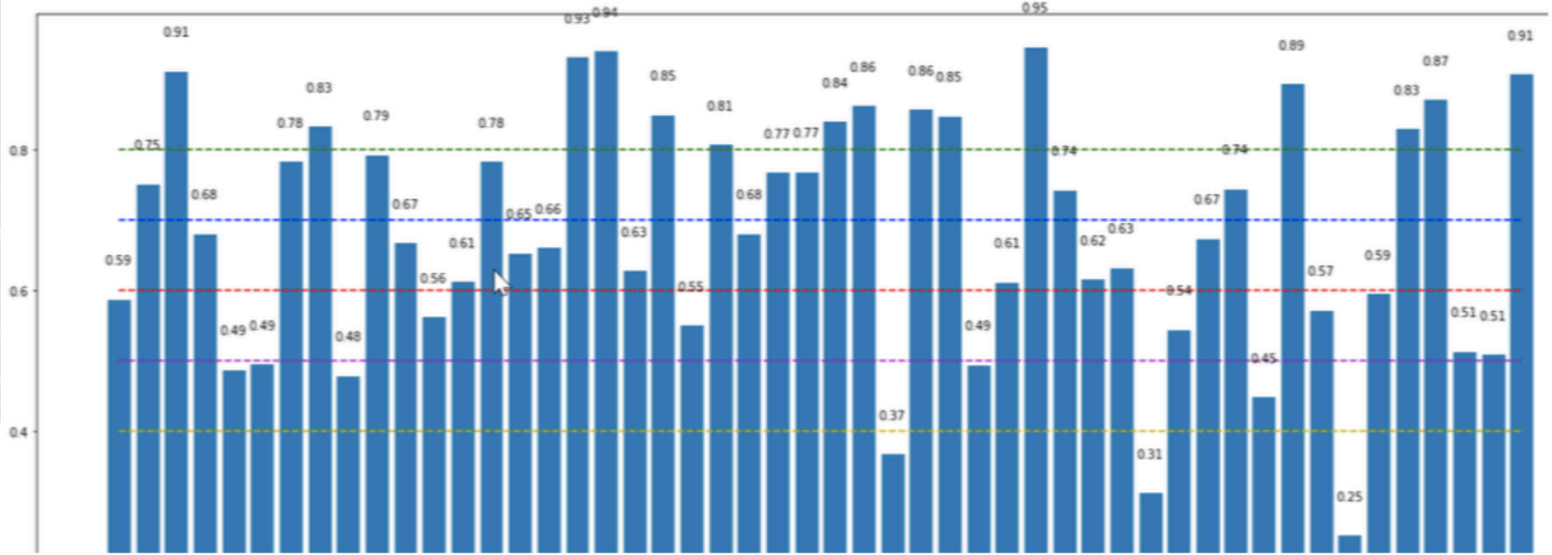
Case Study:
ESC-50 (Environmental Sound Classification)

ConvNet (**Pytorch 1.6.0+cu101**)

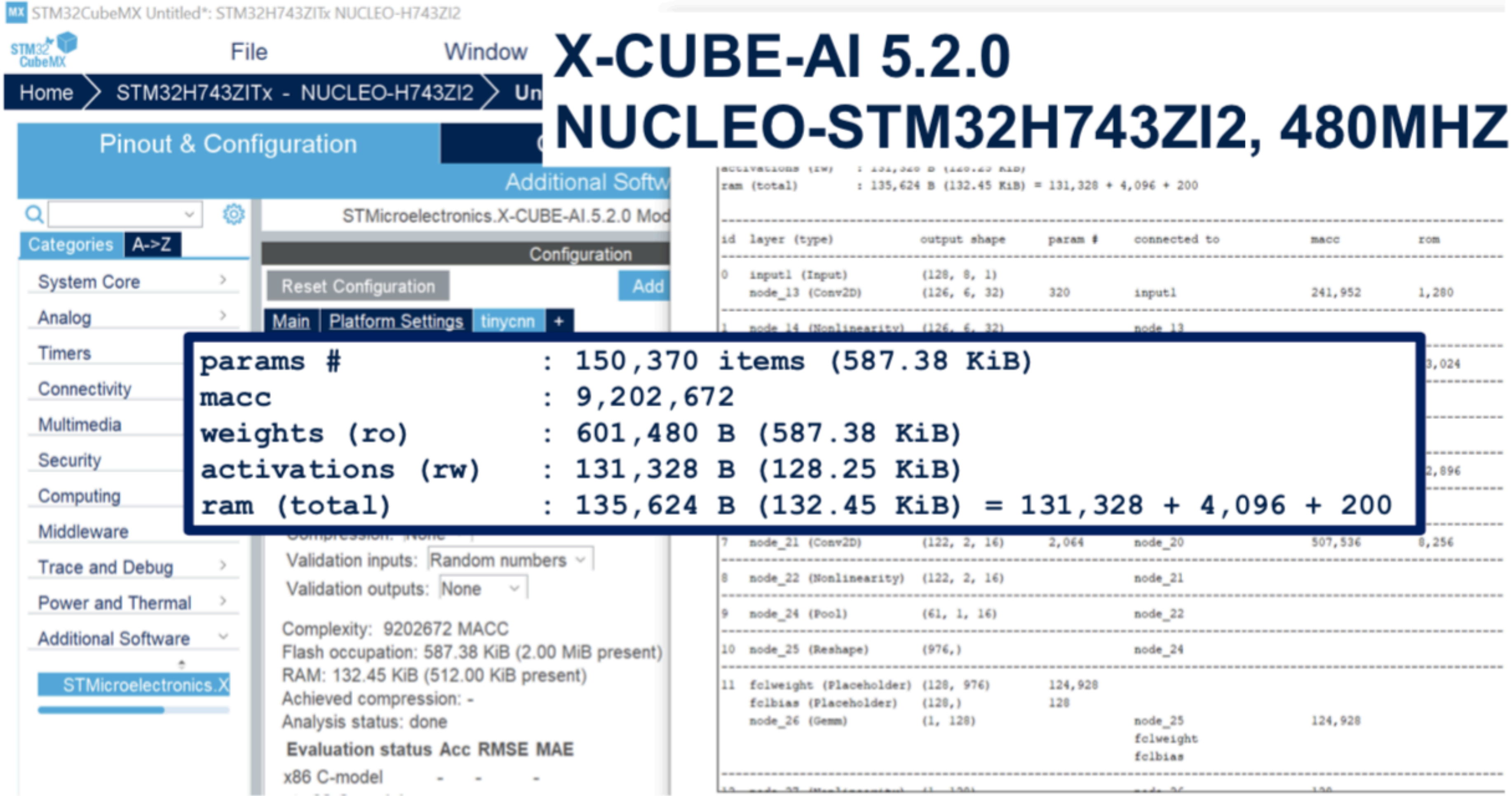


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [1, 32, 126, 6]             320
              ReLU-2            [1, 32, 126, 6]               0
            Conv2d-3            [1, 64, 125, 5]           8,256
              ReLU-4            [1, 64, 125, 5]               0
         AvgPool2d-5            [1, 64, 124, 4]               0
            Conv2d-6            [1, 32, 123, 3]           8,224
              ReLU-7
            Conv2d-8
              ReLU-9
        AvgPool2d-10
          Flatten-11
           Linear-12
           Linear-13
=========================
Total params: 150,370
Trainable params: 150,370
Non-trainable params: 0
-------------------------
Input size (MB): 0.00
Forward/backward pass siz
Params size (MB): 0.57
Estimated Total Size (MB)
-------------------------
```

- 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



Compression: None

Validation inputs: Random numbers

Validation outputs: None

Complexity: 9202672 MACC
Flash occupation: 587.38 KiB (2.00 MiB present)
RAM: 132.45 KiB (512.00 KiB present)
Achieved compression: -
Analysis status: done

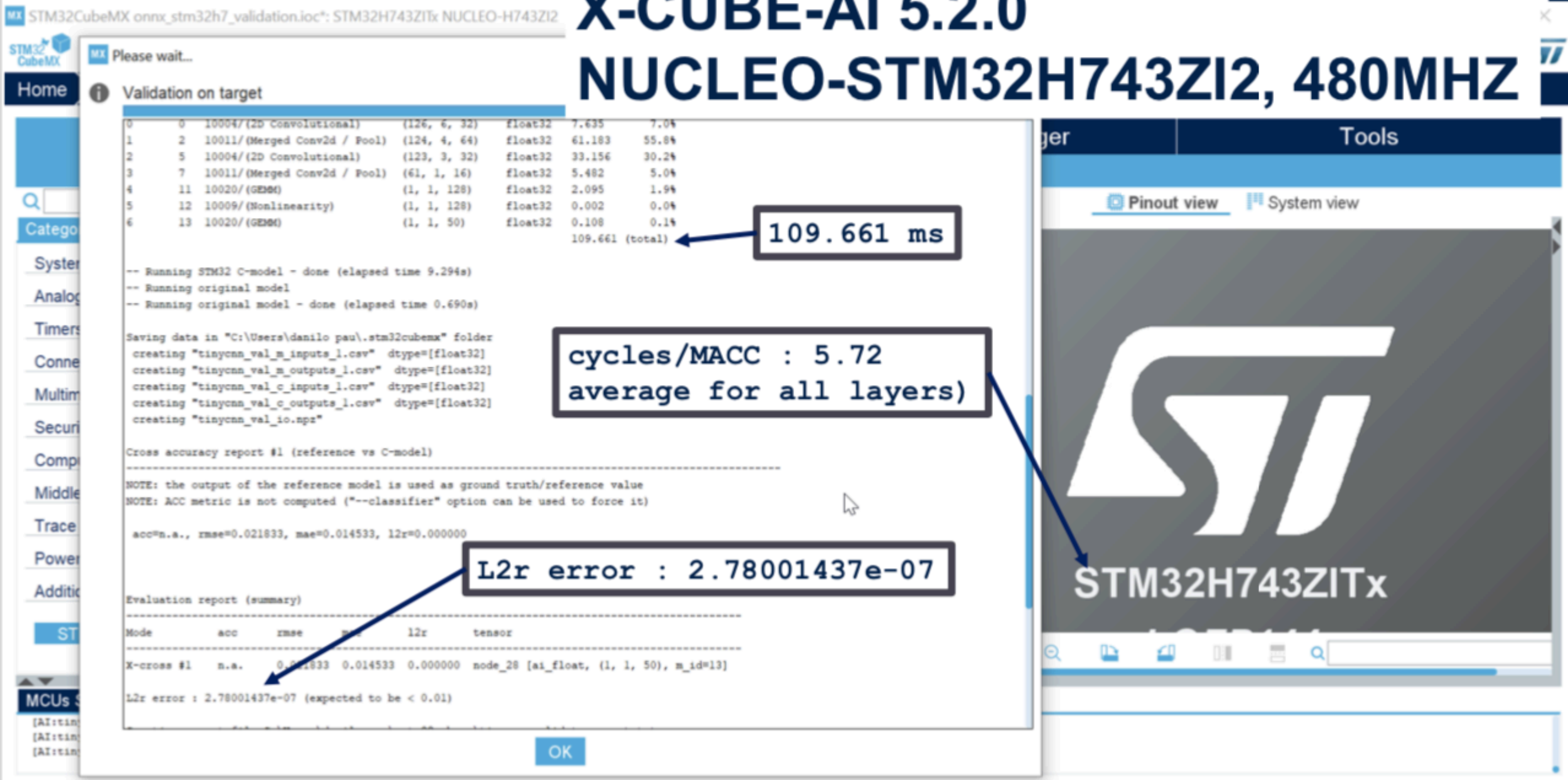
Evaluation status	Acc	RMSE	MAE
x86 C-model	-	-	-

```
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
```

```
activations (rw)    : 131,328 B (128.25 KiB)
ram (total)         : 135,624 B (132.45 KiB) = 131,328 + 4,096 + 200

------------------------------------------------------------------------------------------------------------
id  layer (type)              output shape    param #    connected to              macc        rom
------------------------------------------------------------------------------------------------------------
0   input1 (Input)            (128, 8, 1)
    node_13 (Conv2D)          (126, 6, 32)    320        input1                    241,952     1,280
------------------------------------------------------------------------------------------------------------
1   node_14 (Nonlinearity)    (126, 6, 32)               node_13
------------------------------------------------------------------------------------------------------------
7   node_21 (Conv2D)          (122, 2, 16)    2,064      node_20                   507,536     8,256
------------------------------------------------------------------------------------------------------------
8   node_22 (Nonlinearity)    (122, 2, 16)               node_21
------------------------------------------------------------------------------------------------------------
9   node_24 (Pool)            (61, 1, 16)                node_22
------------------------------------------------------------------------------------------------------------
10  node_25 (Reshape)         (976,)                     node_24
------------------------------------------------------------------------------------------------------------
11  fc1weight (Placeholder)   (128, 976)      124,928
    fc1bias (Placeholder)     (128,)          128
    node_26 (Gemm)            (1, 128)                   node_25                   124,928
                                                         fc1weight
                                                         fc1bias
------------------------------------------------------------------------------------------------------------
```

life.augmented

X-CUBE-AI 5.2.0
NUCLEO-STM32H743ZI2, 480MHZ



```
0       0   10004/(2D Convolutional)      (126, 6, 32)    float32  7.635      7.0%
1       2   10011/(Merged Conv2d / Pool)  (124, 4, 64)    float32  61.183    55.8%
2       5   10004/(2D Convolutional)      (123, 3, 32)    float32  33.156    30.2%
3       7   10011/(Merged Conv2d / Pool)  (61, 1, 16)     float32  5.482      5.0%
4       11  10020/(GEMM)                  (1, 1, 128)     float32  2.095      1.9%
5       12  10009/(Nonlinearity)          (1, 1, 128)     float32  0.002      0.0%
6       13  10020/(GEMM)                  (1, 1, 50)      float32  0.108      0.1%
                                                                   109.661 (total)

-- Running STM32 C-model - done (elapsed time 9.294s)
-- Running original model
-- Running original model - done (elapsed time 0.690s)

Saving data in "C:\Users\danilo pau\.stm32cubemx" folder
 creating "tinycnn_val_m_inputs_1.csv"  dtype=[float32]
 creating "tinycnn_val_m_outputs_1.csv"  dtype=[float32]
 creating "tinycnn_val_c_inputs_1.csv"  dtype=[float32]
 creating "tinycnn_val_c_outputs_1.csv"  dtype=[float32]
 creating "tinycnn_val_io.npz"

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=n.a., rmse=0.021833, mae=0.014533, l2r=0.000000



Evaluation report (summary)
-----------------------------------------------------------------------------------------------
Mode          acc      rmse      mae       l2r       tensor
-----------------------------------------------------------------------------------------------
X-cross #1    n.a.     0.021833  0.014533  0.000000  node_28 [ai_float, (1, 1, 50), m_id=13]

L2r error : 2.78001437e-07 (expected to be < 0.01)
```

109.661 ms

cycles/MACC : 5.72
average for all layers)

L2r error : 2.78001437e-07

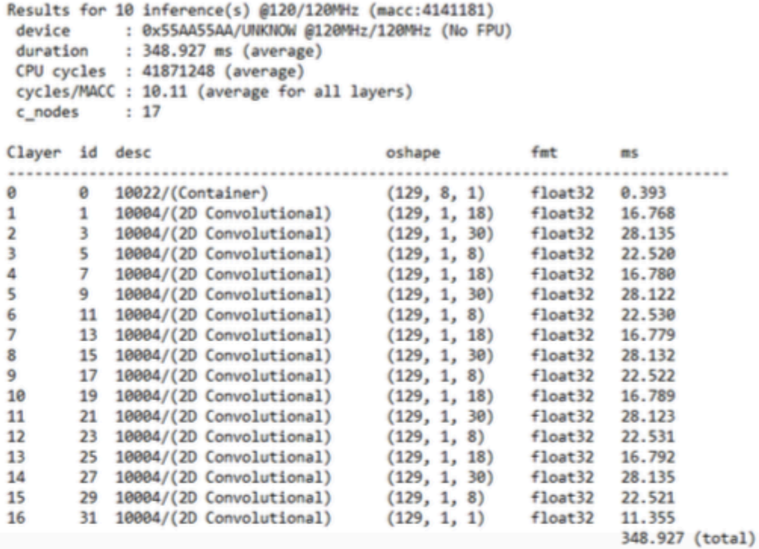
Case Study: Speech Denoise

https://it.mathworks.com/help/deeplearning/ug/denoise-speech-using-deep-learning-networks.html
https://it.mathworks.com/matlabcentral/fileexchange/67296-deep-learning-toolbox-converter-for-onnx-model-format

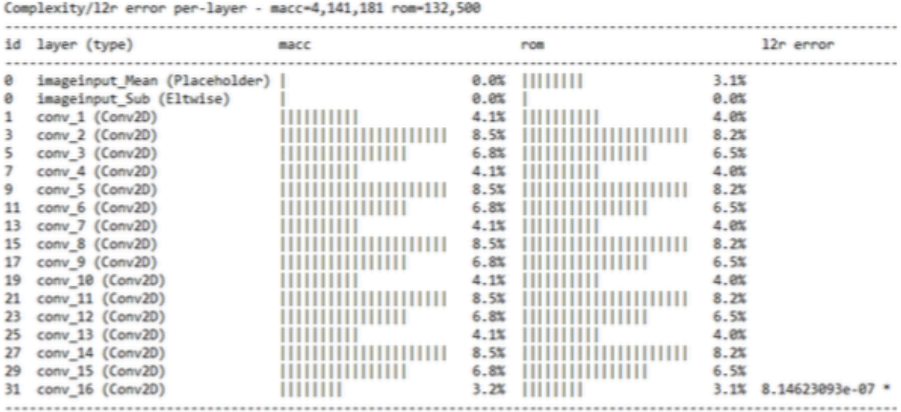
```
params #          : 33,125 items (129.39 KiB)
macc              : 4,141,181
weights (ro)      : 132,500 B (129.39 KiB)
activations (rw)  : 16,152 B (15.77 KiB)
ram (total)       : 20,796 B (20.31 KiB) = 16,152 + 4,128 + 516
```

SPC5-AI v.2.0.0

SPC584B, 120MHZ



```
Results for 10 inference(s) @120/120MHz (macc:4141181)
 device       : 0x55AA55AA/UNKNOW @120MHz/120MHz (No FPU)
 duration     : 348.927 ms (average)
 CPU cycles   : 41871248 (average)
 cycles/MACC  : 10.11 (average for all layers)
 c_nodes      : 17

Clayer  id  desc                        oshape          fmt       ms
--------------------------------------------------------------------------------
0       0   10022/(Container)           (129, 8, 1)     float32   0.393
1       1   10004/(2D Convolutional)    (129, 1, 18)    float32   16.768
2       3   10004/(2D Convolutional)    (129, 1, 30)    float32   28.135
3       5   10004/(2D Convolutional)    (129, 1, 8)     float32   22.520
4       7   10004/(2D Convolutional)    (129, 1, 18)    float32   16.780
5       9   10004/(2D Convolutional)    (129, 1, 30)    float32   28.122
6       11  10004/(2D Convolutional)    (129, 1, 8)     float32   22.530
7       13  10004/(2D Convolutional)    (129, 1, 18)    float32   16.779
8       15  10004/(2D Convolutional)    (129, 1, 30)    float32   28.132
9       17  10004/(2D Convolutional)    (129, 1, 8)     float32   22.522
10      19  10004/(2D Convolutional)    (129, 1, 18)    float32   16.789
11      21  10004/(2D Convolutional)    (129, 1, 30)    float32   28.123
12      23  10004/(2D Convolutional)    (129, 1, 8)     float32   22.531
13      25  10004/(2D Convolutional)    (129, 1, 18)    float32   16.792
14      27  10004/(2D Convolutional)    (129, 1, 30)    float32   28.135
15      29  10004/(2D Convolutional)    (129, 1, 8)     float32   22.521
16      31  10004/(2D Convolutional)    (129, 1, 1)     float32   11.355
                                                                  348.927 (total)
```

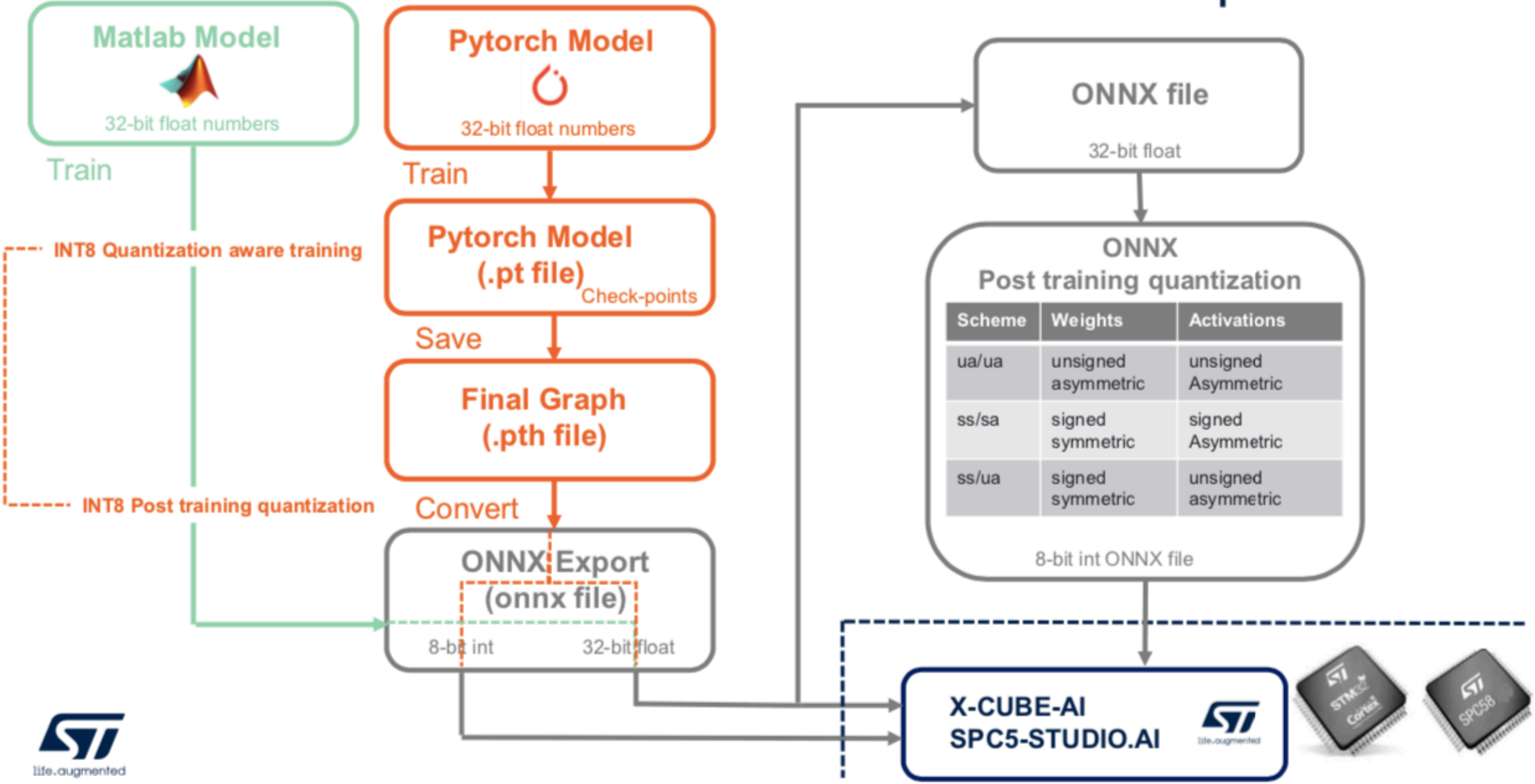


```
Complexity/l2r error per-layer - macc=4,141,181 rom=132,500
------------------------------------------------------------------------------------------------------
id  layer (type)                  macc                               rom                                l2r error
------------------------------------------------------------------------------------------------------
0   imageinput_Mean (Placeholder) |                           0.0%   ||||||||                   3.1%
0   imageinput_Sub (Eltwise)      |                           0.0%   |                          0.0%
1   conv_1 (Conv2D)               ||||||||||                  4.1%   ||||||||||                 4.0%
3   conv_2 (Conv2D)               |||||||||||||||||||||       8.5%   |||||||||||||||||||||      8.2%
5   conv_3 (Conv2D)               ||||||||||||||||            6.8%   ||||||||||||||||           6.5%
7   conv_4 (Conv2D)               ||||||||||                  4.1%   ||||||||||                 4.0%
9   conv_5 (Conv2D)               |||||||||||||||||||||       8.5%   |||||||||||||||||||||      8.2%
11  conv_6 (Conv2D)               ||||||||||||||||            6.8%   ||||||||||||||||           6.5%
13  conv_7 (Conv2D)               ||||||||||                  4.1%   ||||||||||                 4.0%
15  conv_8 (Conv2D)               |||||||||||||||||||||       8.5%   |||||||||||||||||||||      8.2%
17  conv_9 (Conv2D)               ||||||||||||||||            6.8%   ||||||||||||||||           6.5%
19  conv_10 (Conv2D)              ||||||||||                  4.1%   ||||||||||                 4.0%
21  conv_11 (Conv2D)              |||||||||||||||||||||       8.5%   |||||||||||||||||||||      8.2%
23  conv_12 (Conv2D)              ||||||||||||||||            6.8%   ||||||||||||||||           6.5%
25  conv_13 (Conv2D)              ||||||||||                  4.1%   ||||||||||                 4.0%
27  conv_14 (Conv2D)              |||||||||||||||||||||       8.5%   |||||||||||||||||||||      8.2%
29  conv_15 (Conv2D)              ||||||||||||||||            6.8%   ||||||||||||||||           6.5%
31  conv_16 (Conv2D)              ||||||||                    3.2%   ||||||||                   3.1%  8.14623093e-07 *
------------------------------------------------------------------------------------------------------
```

L2r error 8.14623093e-07

Inference time 348.927 ms

Desired Development Flow



Matlab Model

32-bit float numbers

Train

Pytorch Model

32-bit float numbers

Train

Pytorch Model (.pt file)

Check-points

Save

Final Graph (.pth file)

Convert

INT8 Quantization aware training

INT8 Post training quantization

ONNX Export (onnx file)

8-bit int

32-bit float

ONNX file

32-bit float

ONNX Post training quantization

Scheme	Weights	Activations
ua/ua	unsigned asymmetric	unsigned Asymmetric
ss/sa	signed symmetric	signed Asymmetric
ss/ua	signed symmetric	unsigned asymmetric

8-bit int ONNX file

X-CUBE-AI

SPC5-STUDIO.AI

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

life.augmented

11 Slides/9 minutes

BECKHOFF

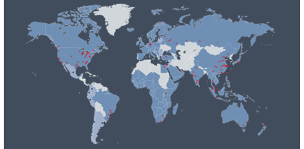
Neural Automation: Fusion of Automation and Data Science

- Speaker: Fabian Bause
- ONNX Community Virtual Workshop

About Beckhoff and myself

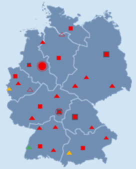
- Fabian Bause
- PhD in Electrical Engineering
- Product Manager TwinCAT at Beckhoff since 01/2016
- Technological responsibilities
  - Machine Learning
  - Integration of MATLAB and Simulink
  - Integration of LabVIEW
  - Signal Processing Libraries

75



Subsidiaries and distributors

Verl, Germany

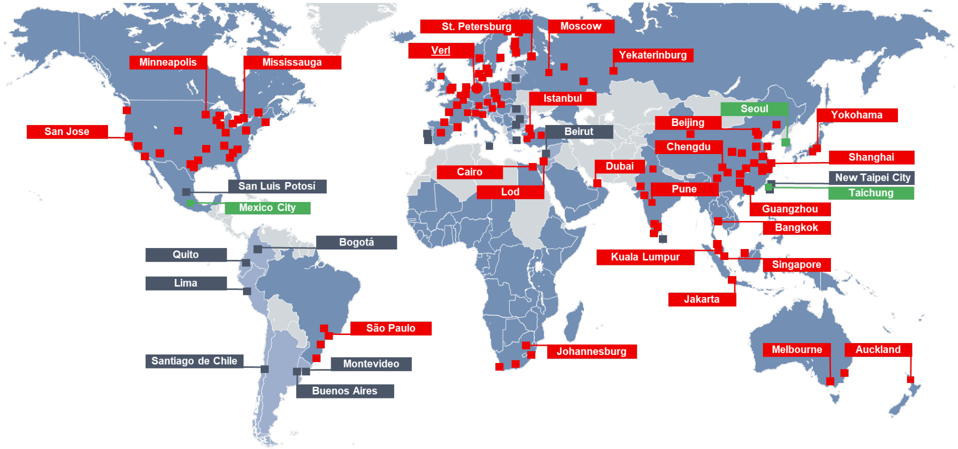


Headquarters

4350

Employees worldwide

Sales worldwide 2019: € 903 million



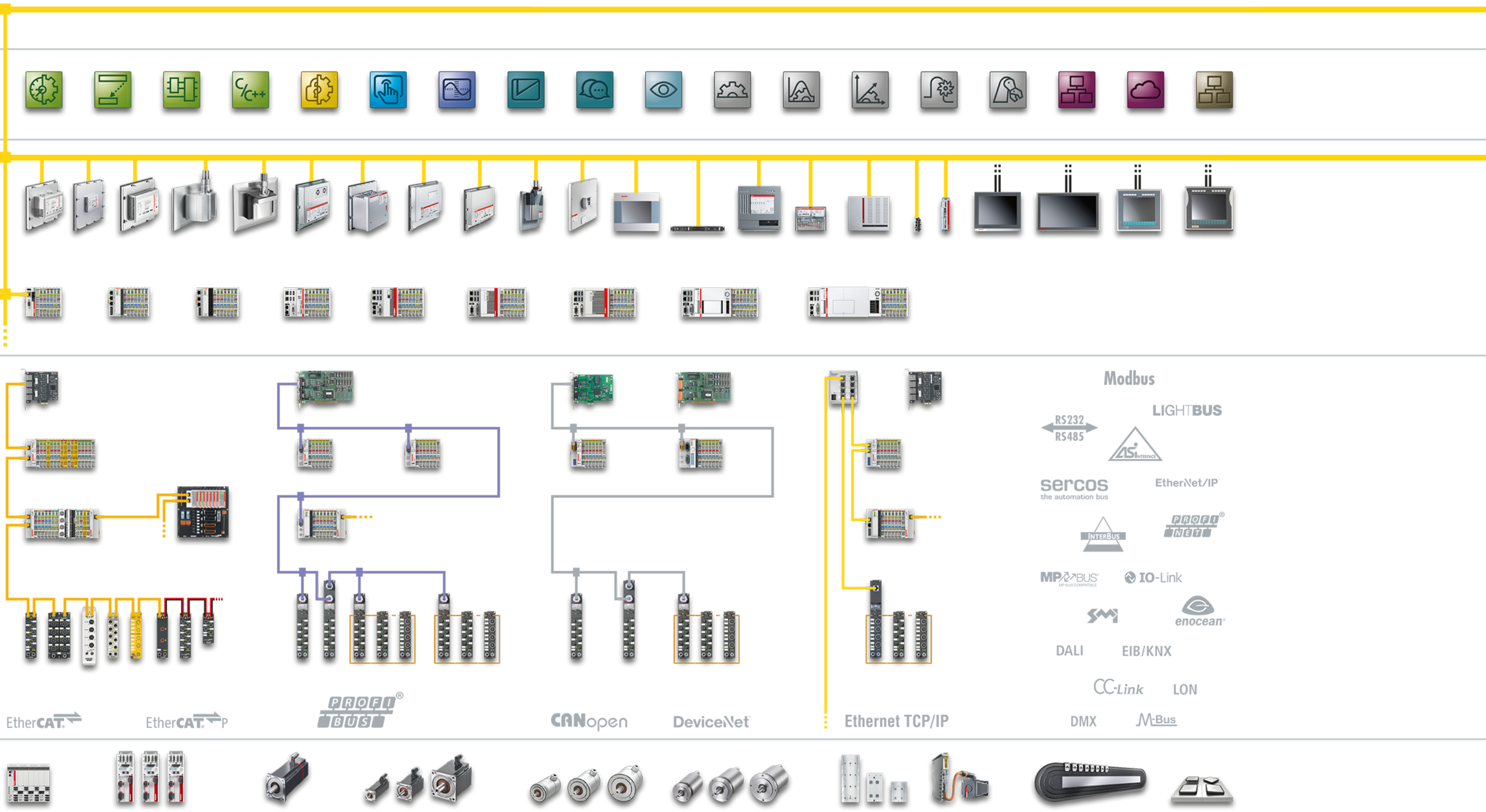
Quick look into the Beckhoff component portfolio

Automation (TwinCAT)

IPC

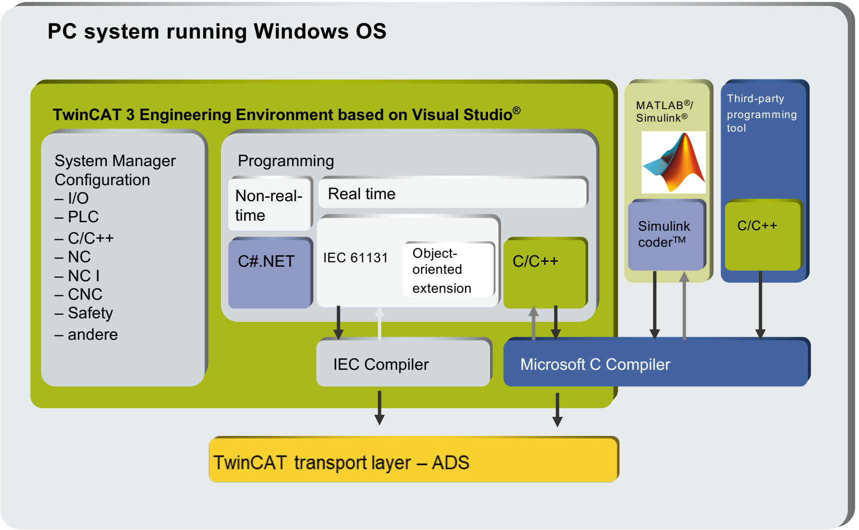
I/O

Motion



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

BECKHOFF



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

C#.NET

Real time

IEC 61131

Object-oriented extension

C/C++

IEC Compiler

Microsoft C Compiler

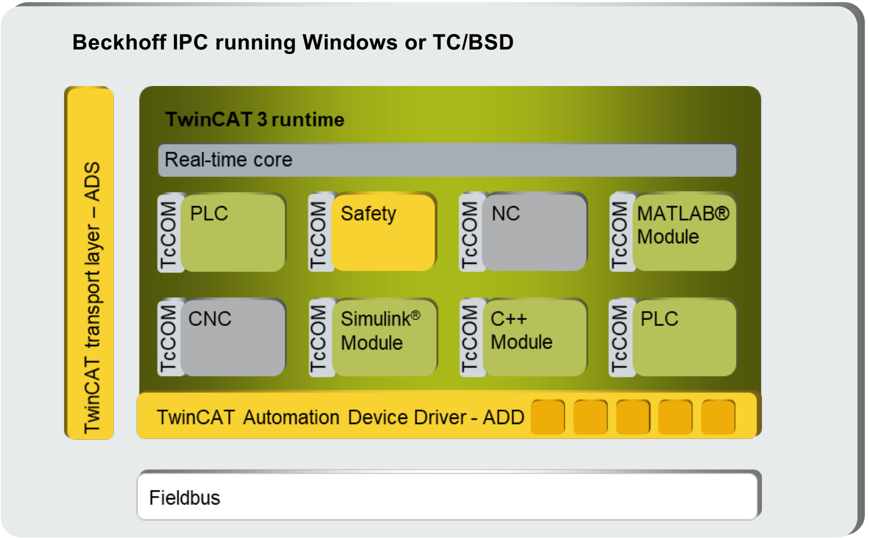
MATLAB®/ Simulink®

Simulink coder™

Third-party programming tool

C/C++

TwinCAT transport layer – ADS



Beckhoff IPC running Windows or TC/BSD

TwinCAT transport layer – ADS

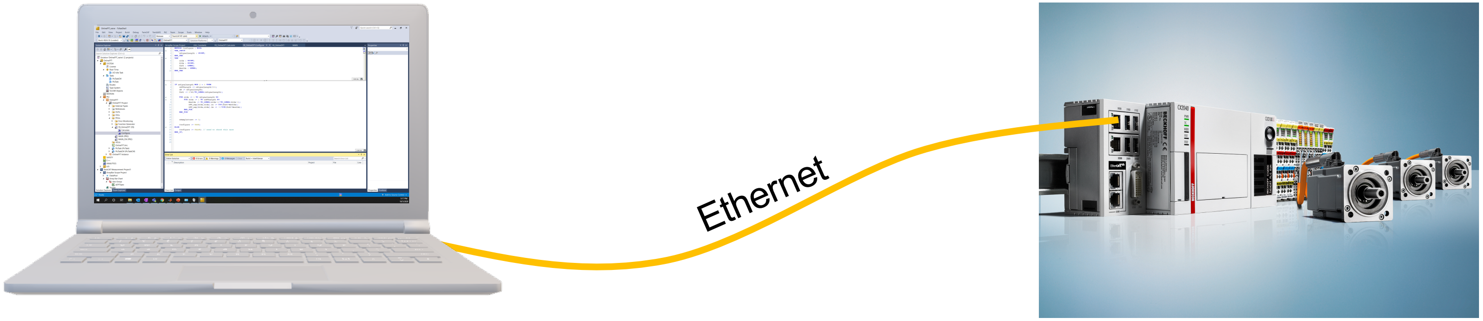
TwinCAT 3 runtime

Real-time core

TcCOM PLC	TcCOM Safety	TcCOM NC	TcCOM MATLAB® Module
TcCOM CNC	TcCOM Simulink® Module	TcCOM C++ Module	TcCOM PLC

TwinCAT Automation Device Driver - ADD

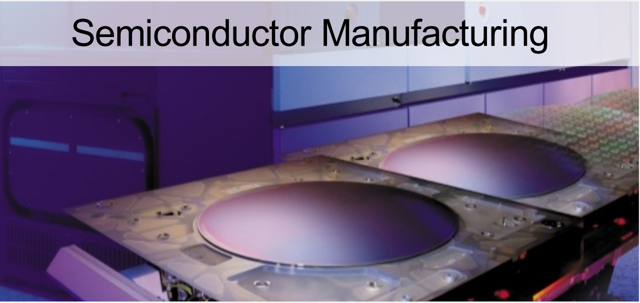
Fieldbus



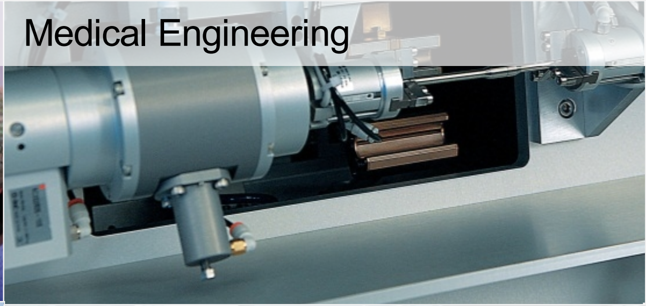
What do Beckhoff customers do with all these components?
They build machines for…

BECKHOFF

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

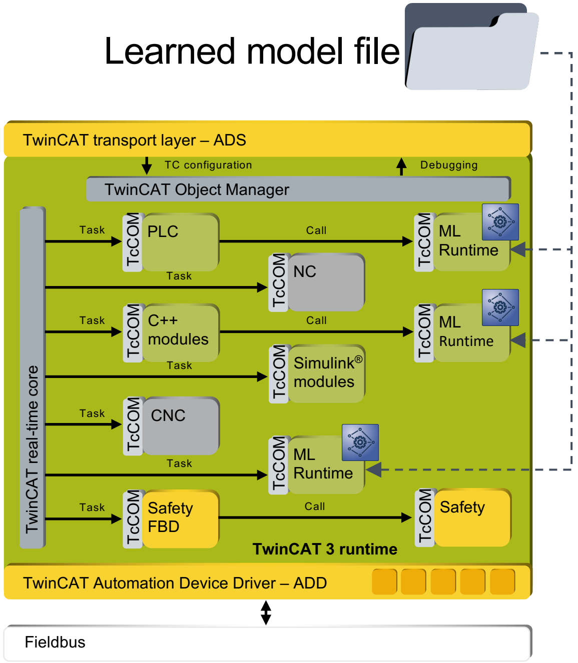
- fast growing standardized file format for ML

Non-blocking parallelization

- parallel use of one TcCOM object by multiple tasks

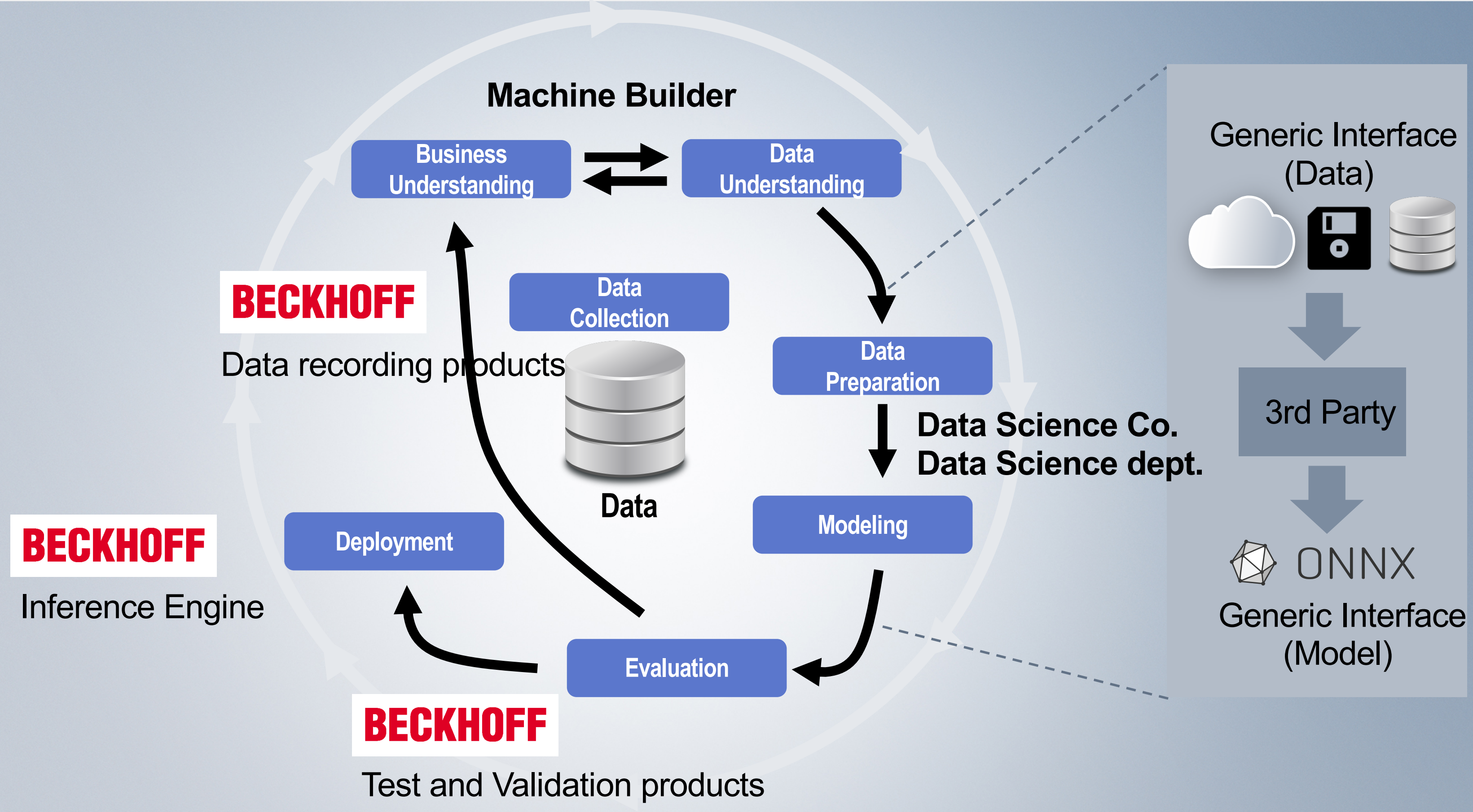
Scalable performance with PC-based control

- Highly optimized performance by using latest SIMD extensions



Why ONNX is important for Beckhoff?

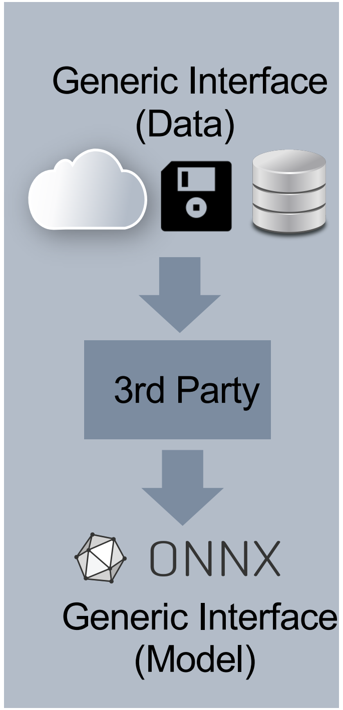
BECKHOFF



Why ONNX is important for Beckhoff?

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



Generic Interface (Data)

3rd Party

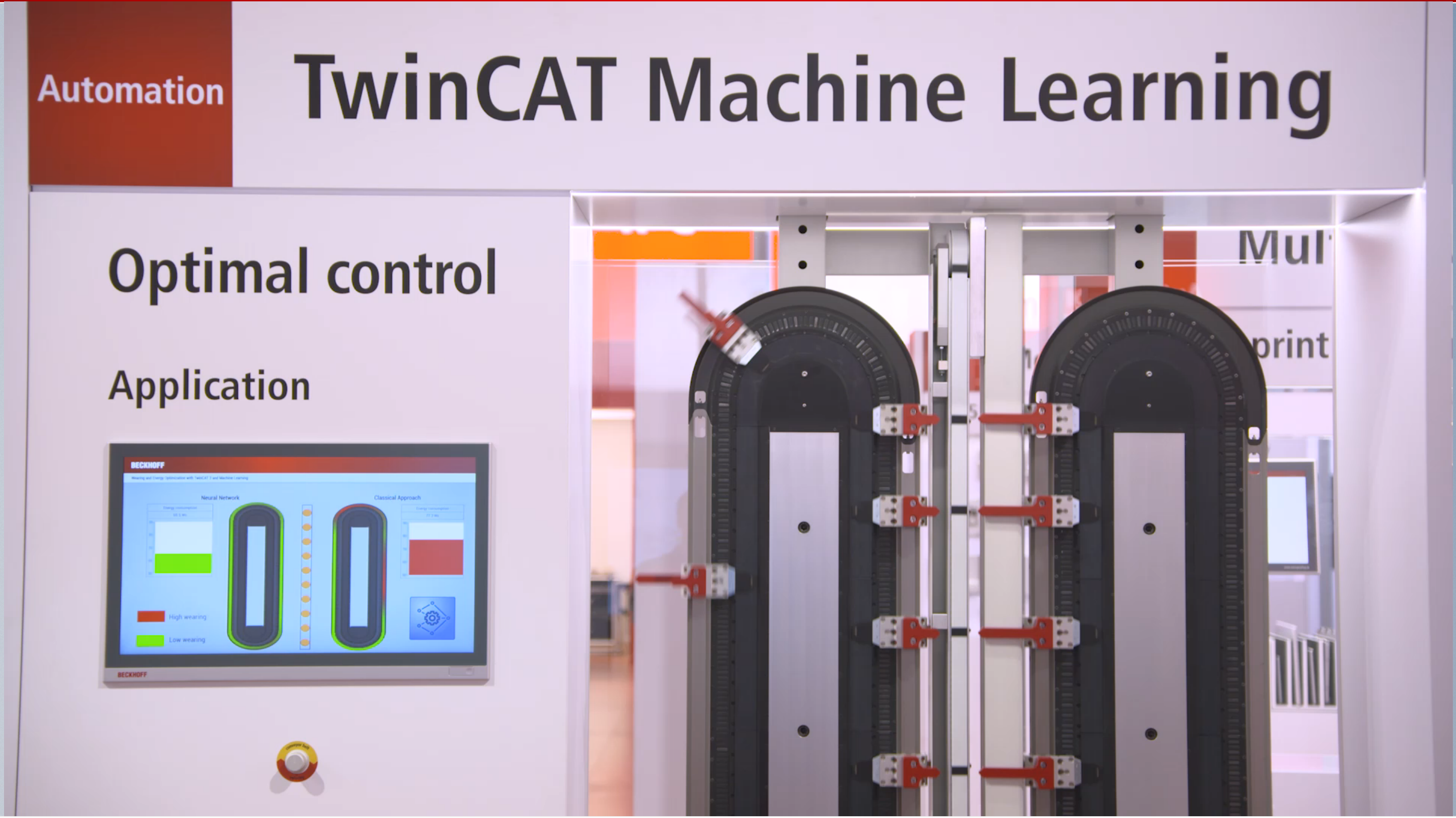
ONNX

Generic Interface (Model)

*see www.beckhoff.ai

Use Case: MLP for optimized motion planning

BECKHOFF



Thank You!

BECKHOFF

Contact: f.bause@beckhoff.com

Contact

BECKHOFF

Beckhoff Automation GmbH & Co. KG
Headquarters
Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
E-mail:	info@beckhoff.com
Web:	www.beckhoff.com

The information provided in this presentation contains merely general descriptions or characteristics of performance which in case of actual application do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressively agreed in the terms of contract.

18 Slides/9 minutes

ONNX RUNTIME

UPDATE (October 2020)

Mobile, Training,
Quantization

Faith Xu | AI Frameworks @ Microsoft

ONNX Runtime 1.5 Release

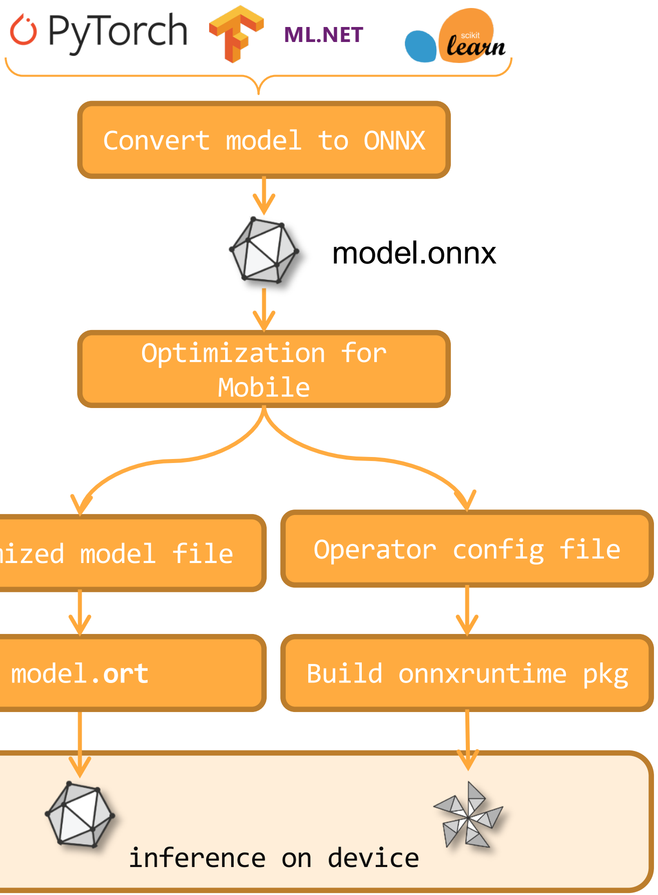
Highlights: Minimal Builds, Training, Quantization

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

model.onnx

Optimization for Mobile

Optimized model file

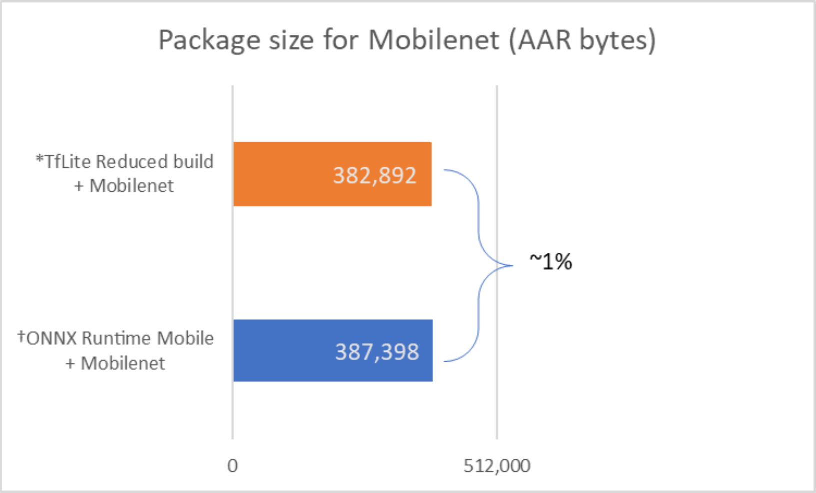
Operator config file

model.ort

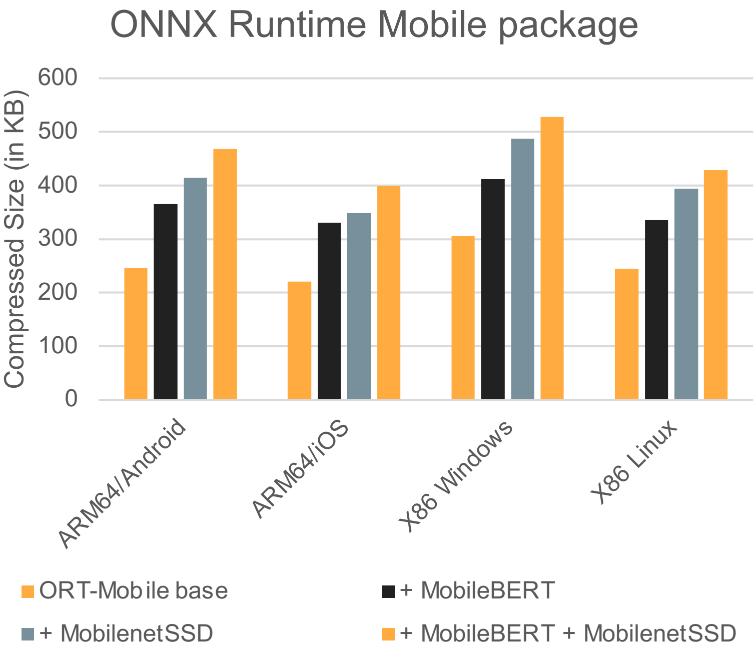
Build onnxruntime pkg

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



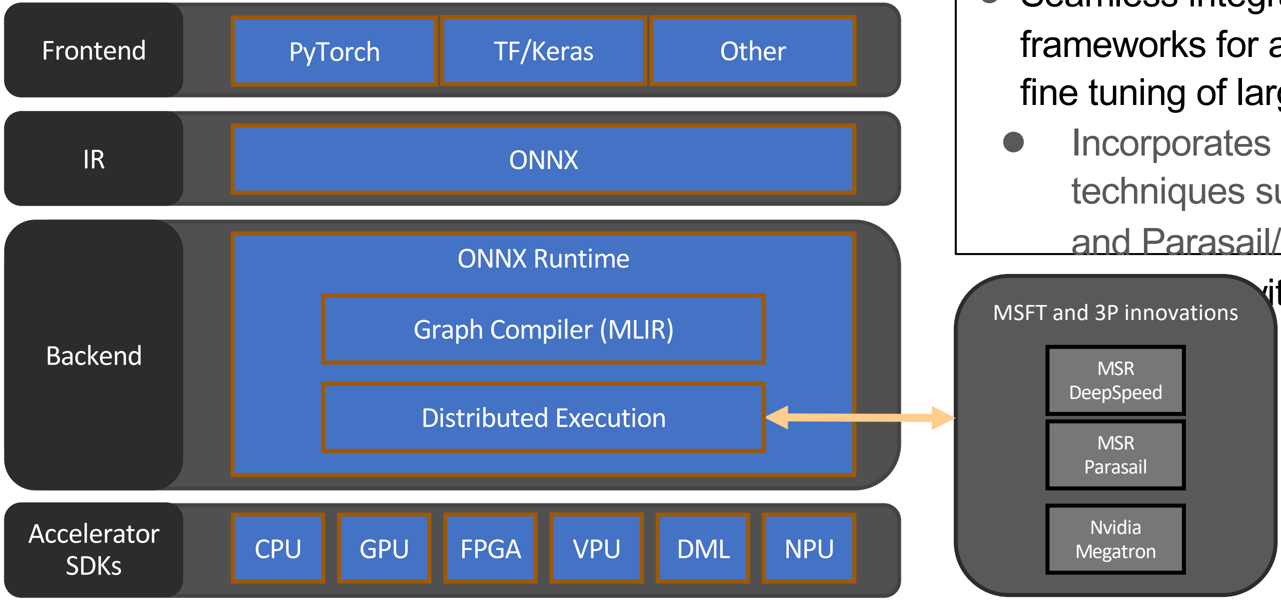
Training Acceleration

Transformer models

ONNX Runtime Training (Public Preview)

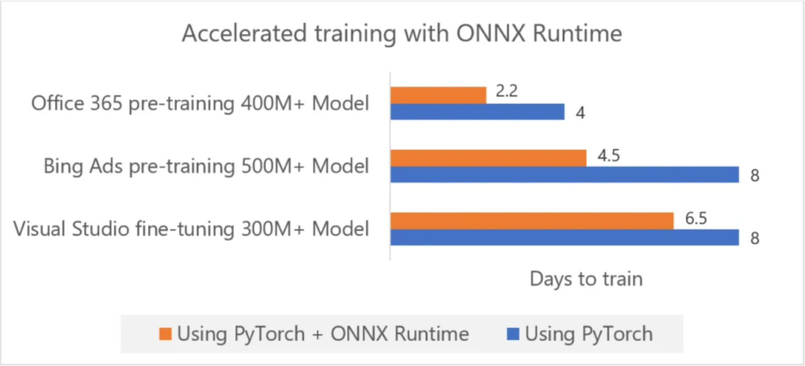
- Seamless integration with existing training frameworks for accelerated training and fine tuning of large transformer models
  - Incorporates latest algorithms and techniques such as DeepSpeed/ZeRO and Parasail/Adasum

ith GPU for distributed



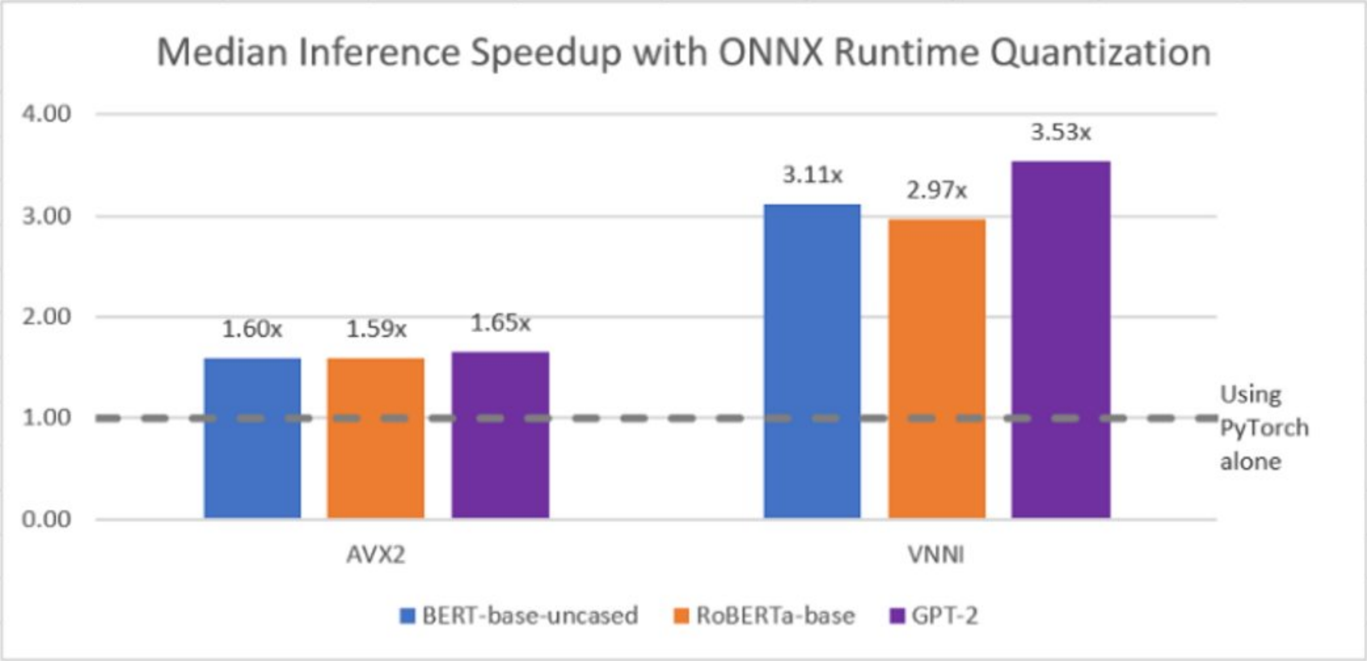
Usage of ORT Training at Microsoft

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

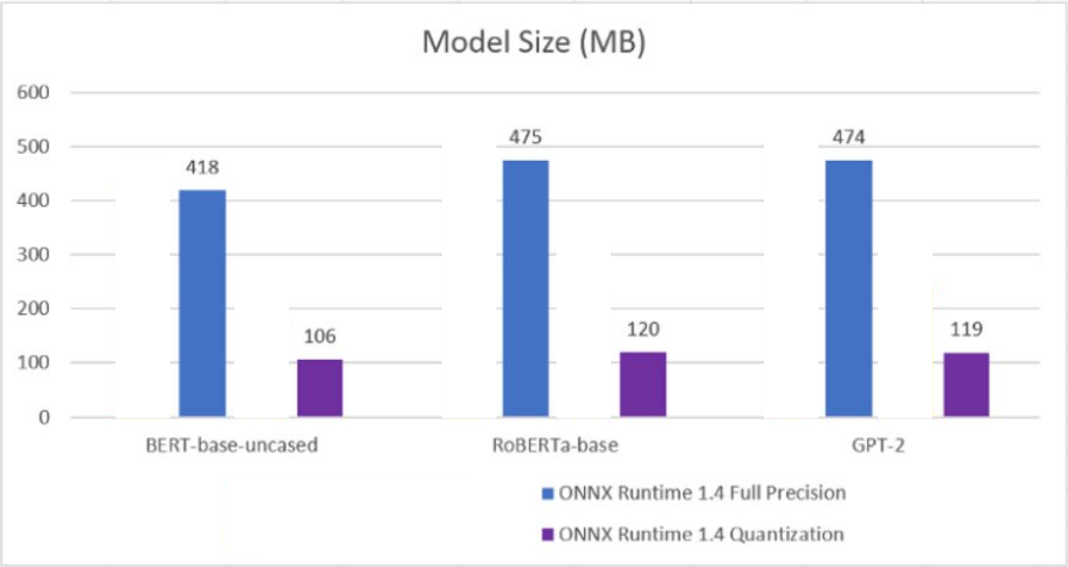


Quantization

Latency improvement

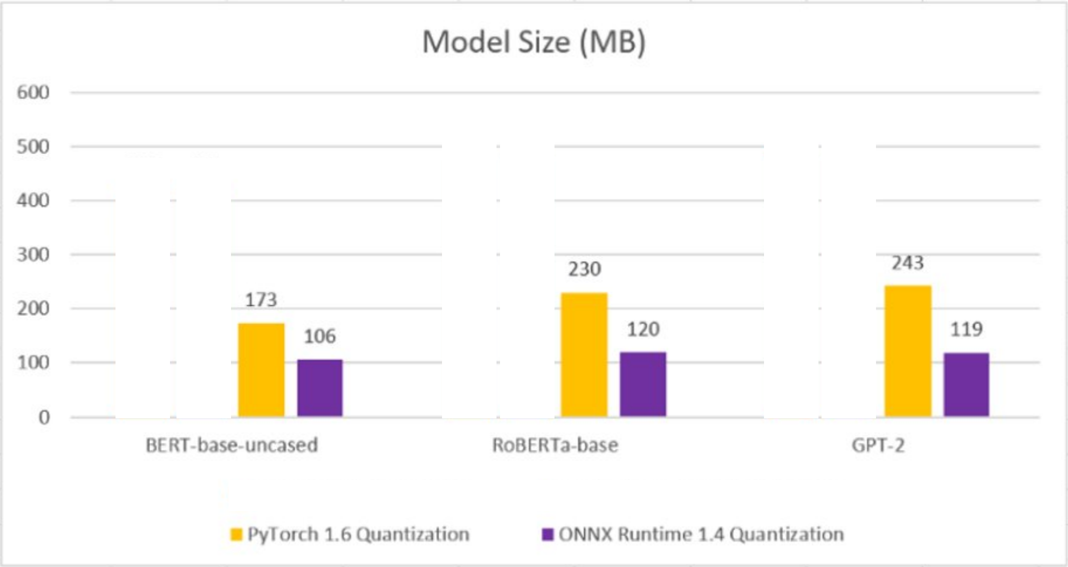


Model size reduction



Int8 quantization for 4x reduction in size

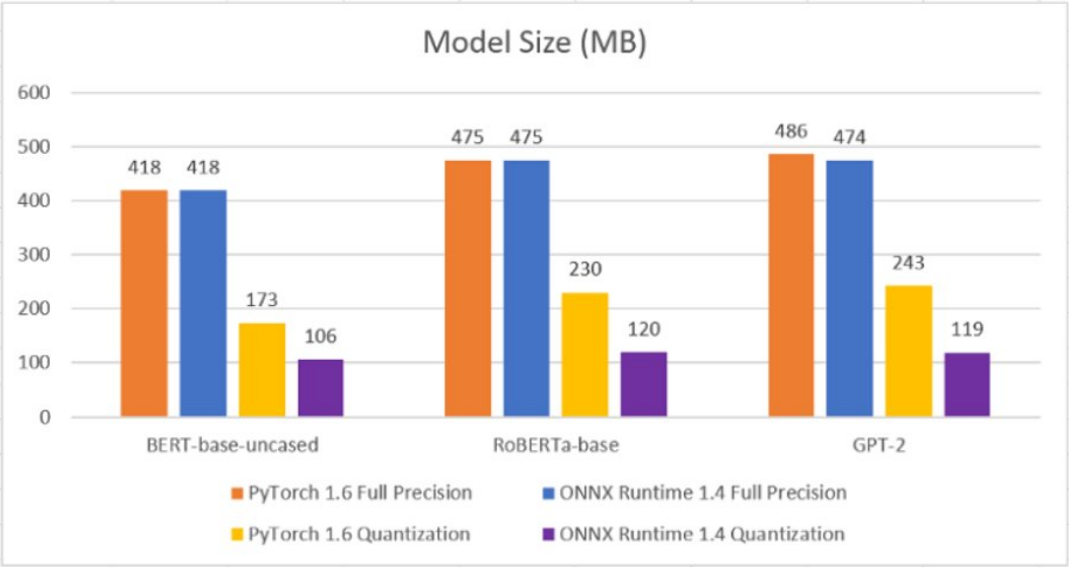
Model size reduction



Int8 quantization for 4x reduction in size

Half the size of quantized PyTorch model

Model size reduction



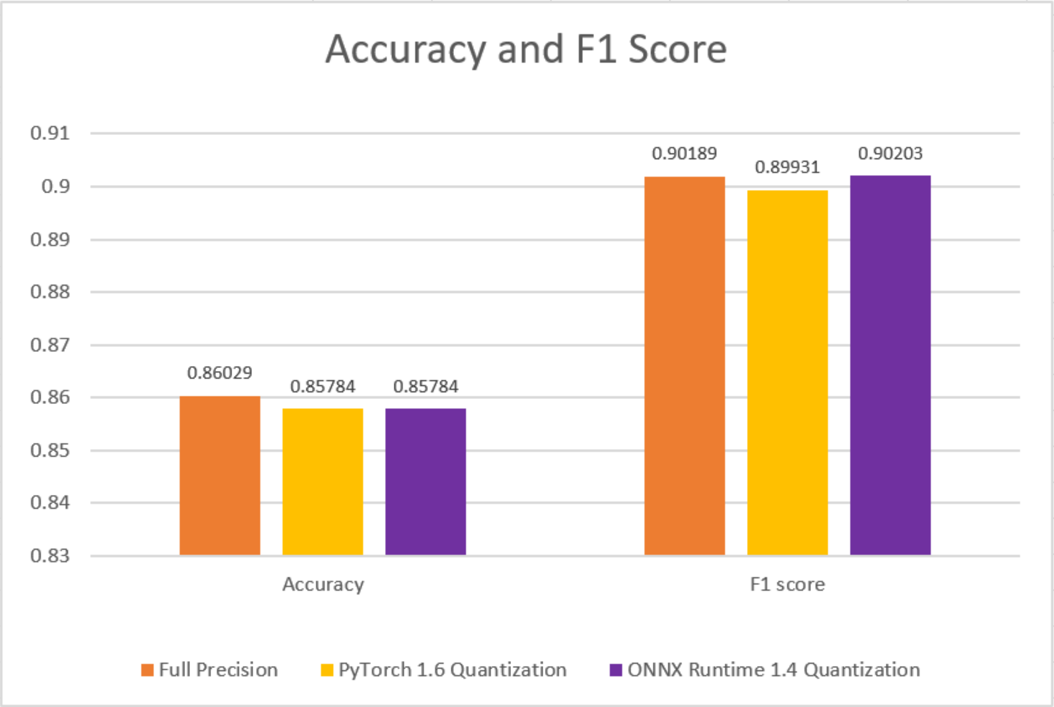
Int8 quantization for 4x reduction in size

Half the size of quantized PyTorch model

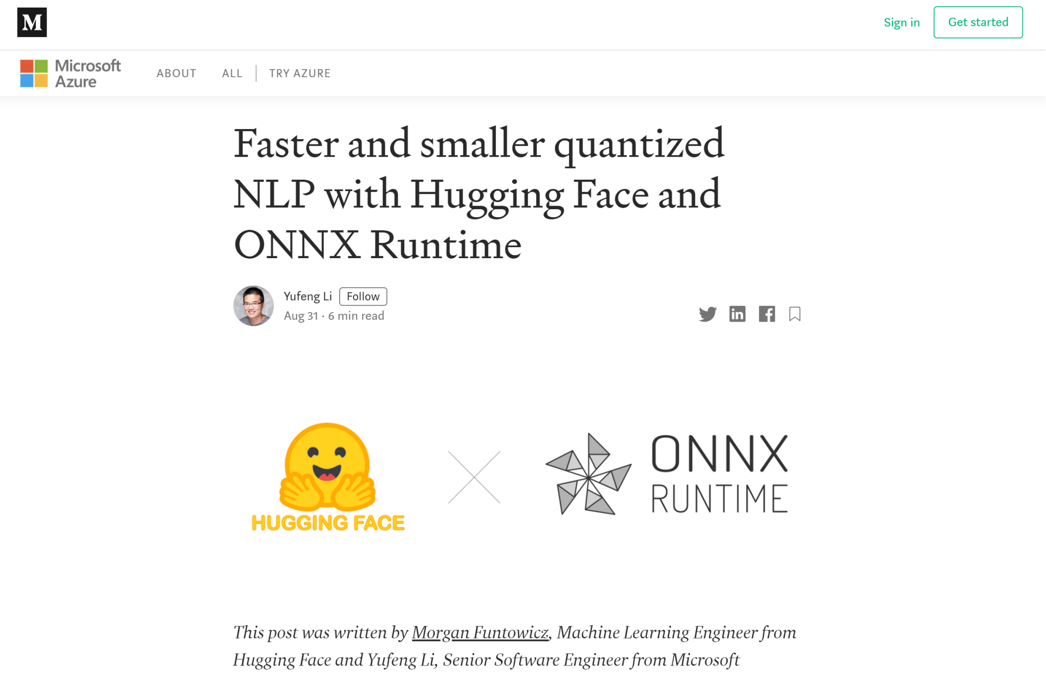
…with minimal accuracy tradeoff

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

Yufeng Li

Aug 31 · 6 min read

This post was written by Morgan 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**

17 Slides/9 minutes

Apache TVM and ONNX

What can ONNX do for DL Compilers (and vice versa)?

Jason Knight - CPO

OctoML

jknight@octoml.ai

Agenda

… and in 10 minutes …
Let’s go!

Intro to TVM

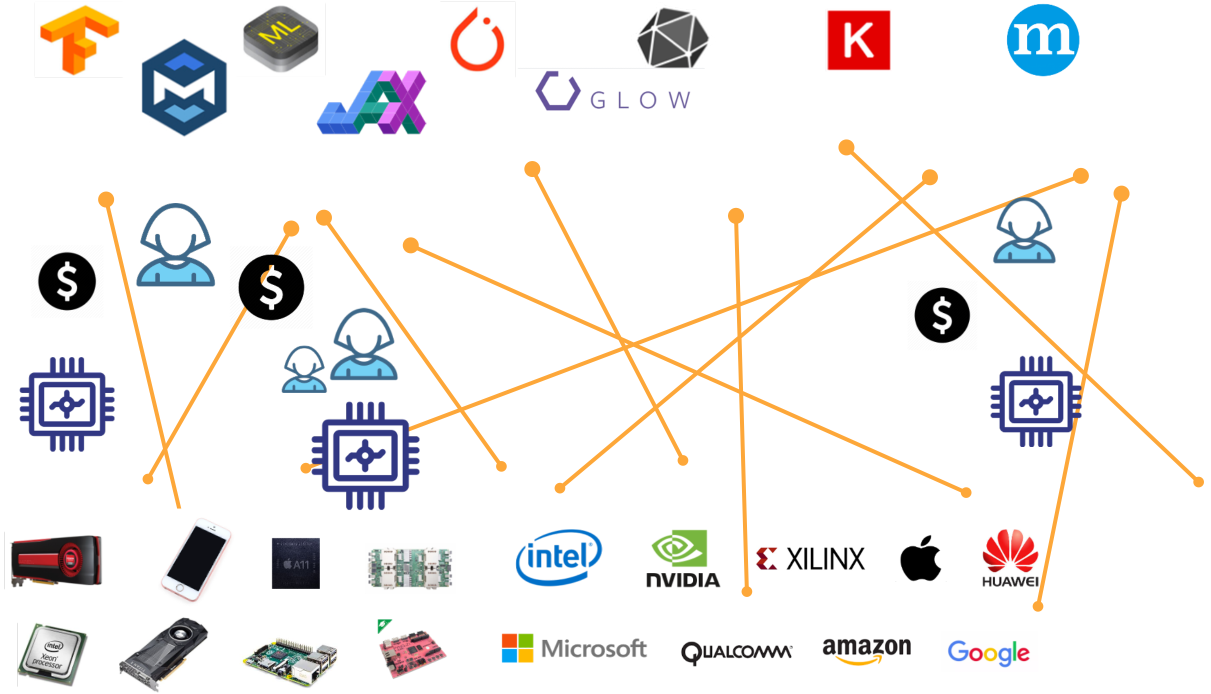
Cool results (TVM + ONNX)

How does it work?

OctoML’s wishlist for ONNX

An exploding ecosystem makes **deployment** painful

Rapidly evolving ML software ecosystem



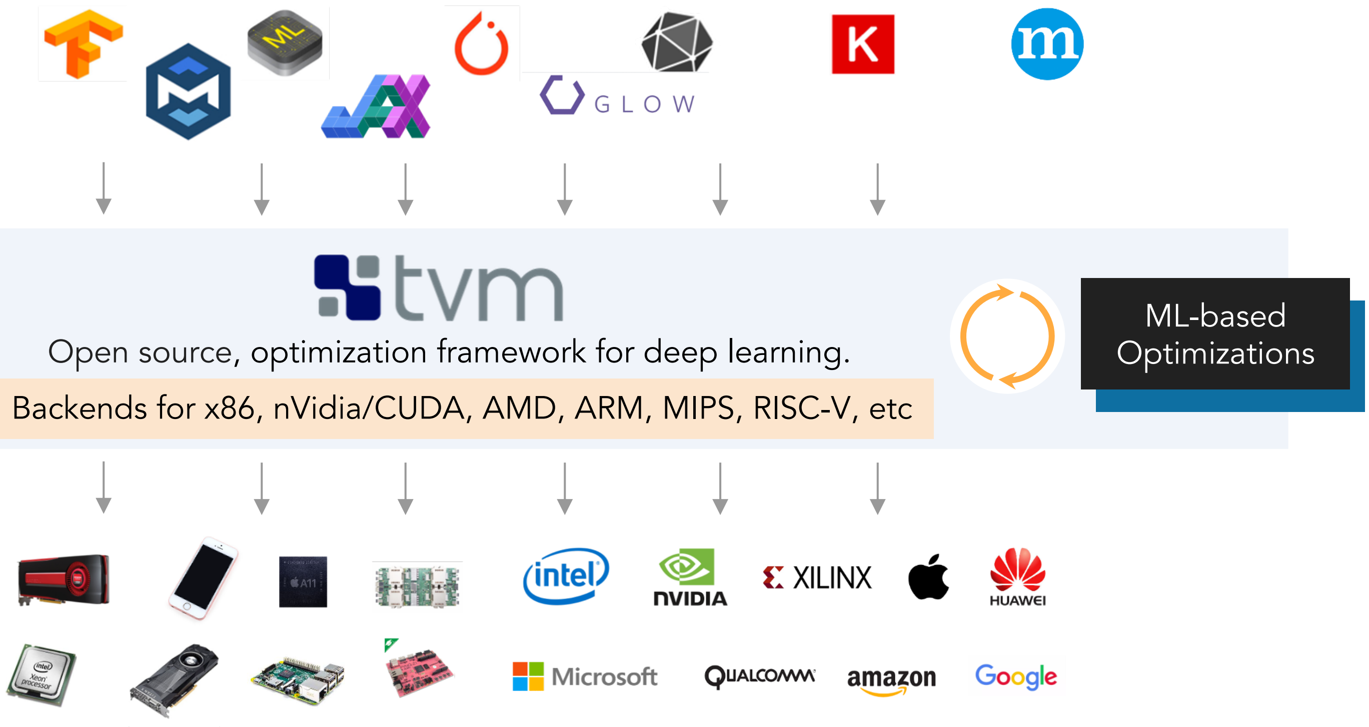
Cambrian explosion of HW backends

TVM: Bridging the gap as a DL compiler and runtime

Reduce model time-to-market

Build your model once, run anywhere

Cut capital and operational ML costs



Open source, optimization framework for deep learning.

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

ML-based Optimizations

TVM is an emerging industry standard ML stack

aws

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

tvm

THE APACHE SOFTWARE FOUNDATION

Open source
~428+ contributors from industry and academia.

"[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.

Microsoft

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

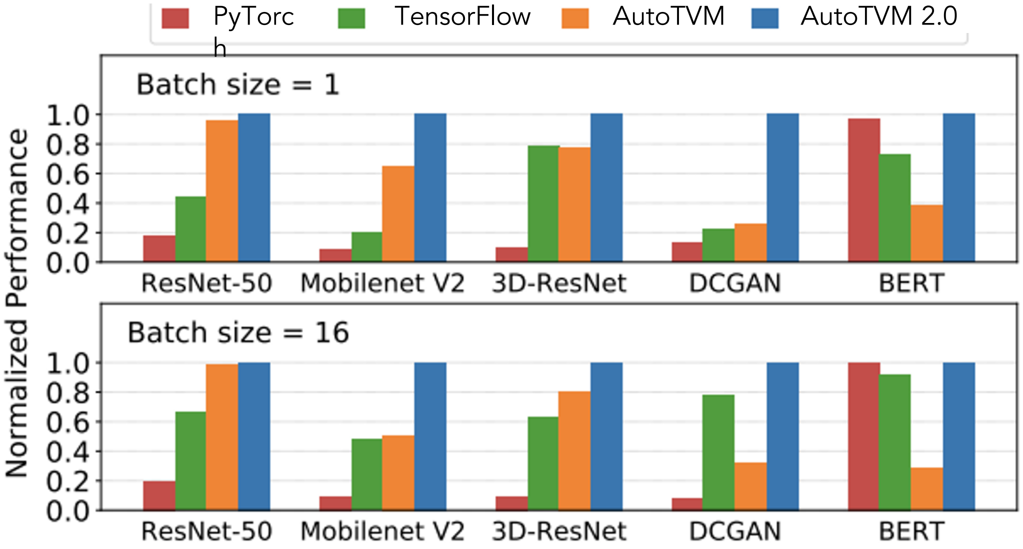
QUALCOMM®

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

ARM AMD XILINX® SiMa.ai™ UNTETHER AI

The power of TVM + ONNX (AKA Results)

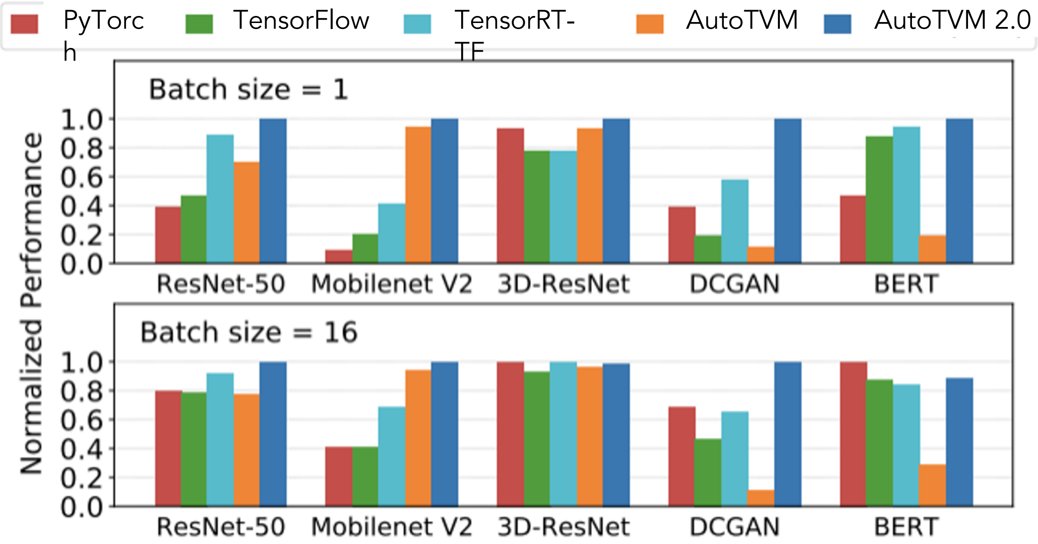
Performance: TVM on x86



(a) Intel CPU

20 core Intel-Platinum-8269CY fp32 performance data from https://arxiv.org/pdf/2006.06762.pdf

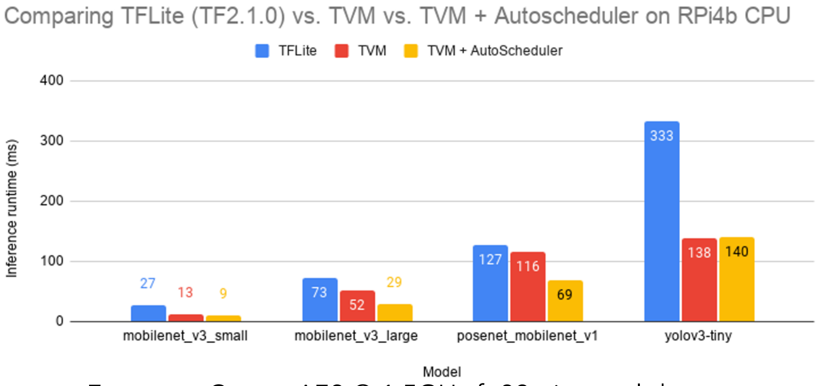
Performance: TVM on GPU



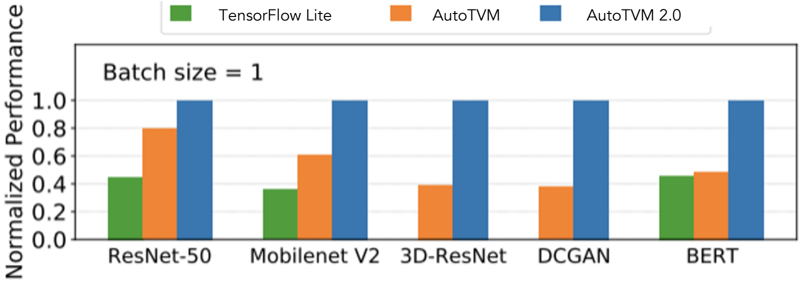
(b) NVIDIA GPU

V100 fp32 performance data from https://arxiv.org/pdf/2006.06762.pdf

Performance: TVM on ARM



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



(c) ARM CPU

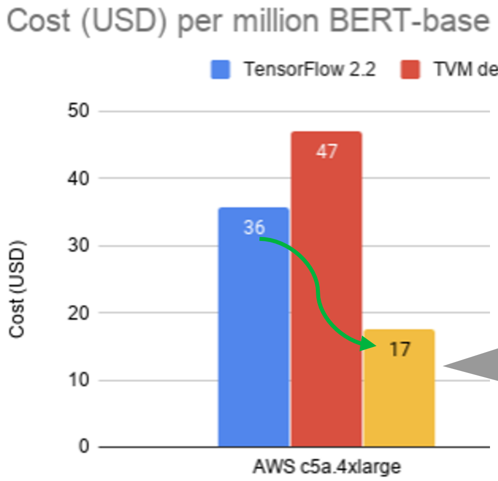
Four core Cortex-A53 @ 1.4GHz fp32 - https://arxiv.org/pdf/2006.06762.pdf

Case Study: 50% reduction in Cloud NLP inference costs

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

Jason Knight
Jul 17 · 5 min read

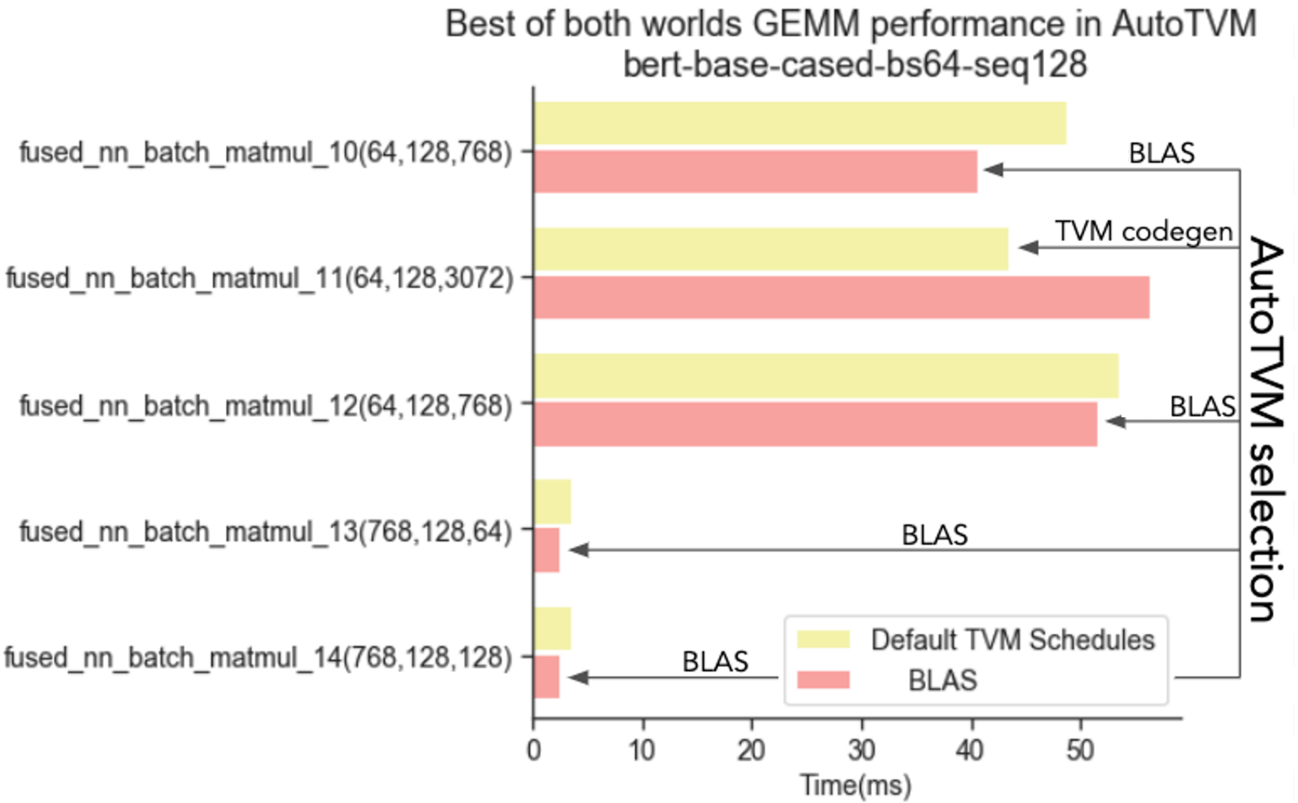
By Joshua Fromm, Bing Xu, Morgan Funtowicz and Jason Knight



Cost (USD) per million BERT-base inferences

2x lower cost on AMD EPYC CPU

Best of both worlds

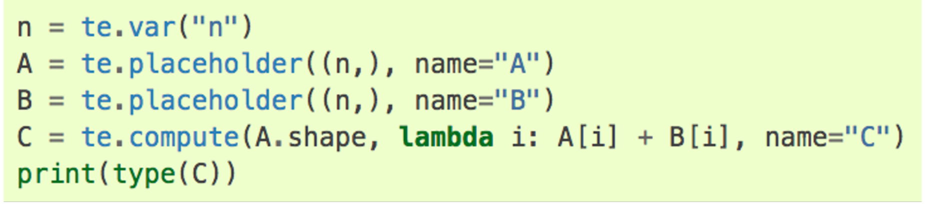


Not enough time!

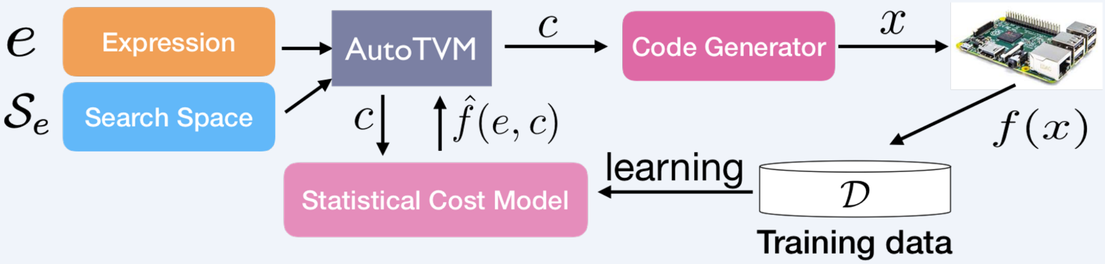
- TensorCore performance (better than cuBLAS)
- Classical ML (better than XGBoost and RAPIDS)
- uTVM for TinyML - ML for microcontrollers
- Int{8,4,3,2,1} and posit quantization support
- ML in your browser - WebGPU and WASM as TVM backends
- … and more!

How does it work?

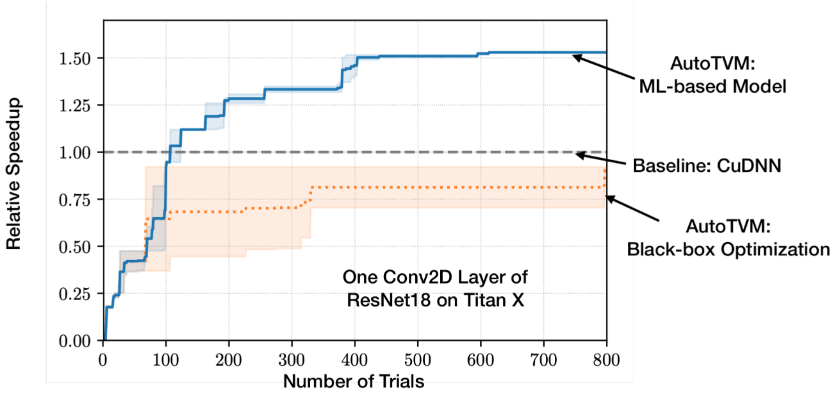
AutoTVM Overview



```
n = te.var("n")
A = te.placeholder((n,), name="A")
B = te.placeholder((n,), name="B")
C = te.compute(A.shape, lambda i: A[i] + B[i], name="C")
print(type(C))
```



Automatically adapt
to hardware type
by learning



10 Slides/9 minutes

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.

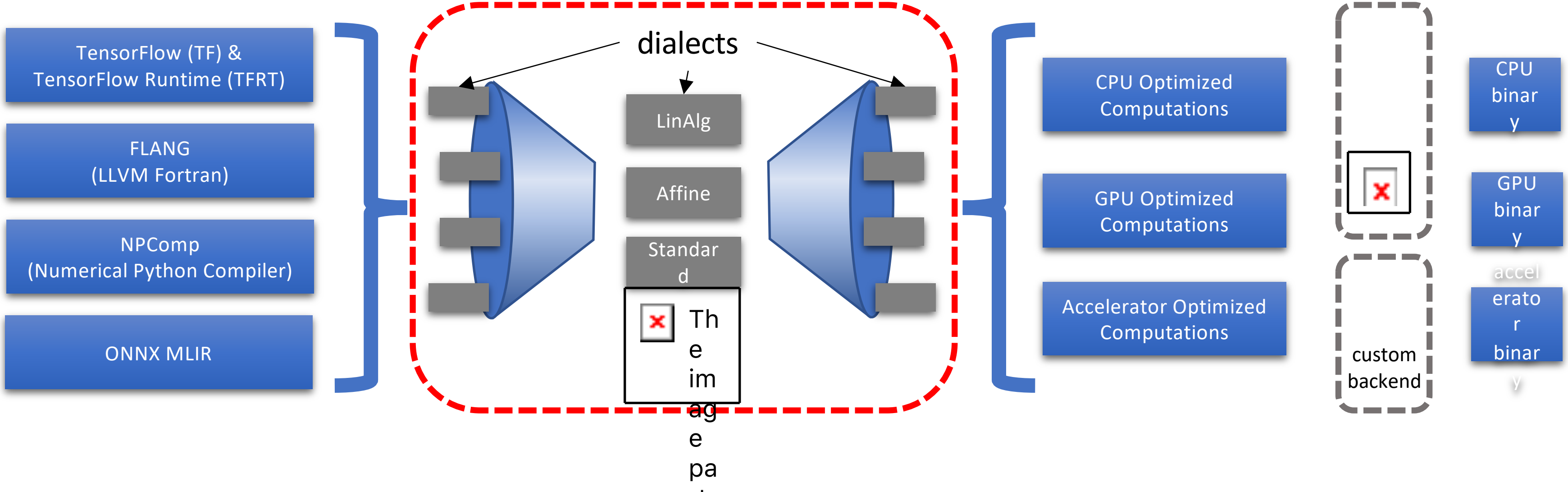
Presenter

Alexandre Eichenberger
Principal Research Staff Member
IBM T.J. Watson Research Center
alexe@us.ibm.com

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.

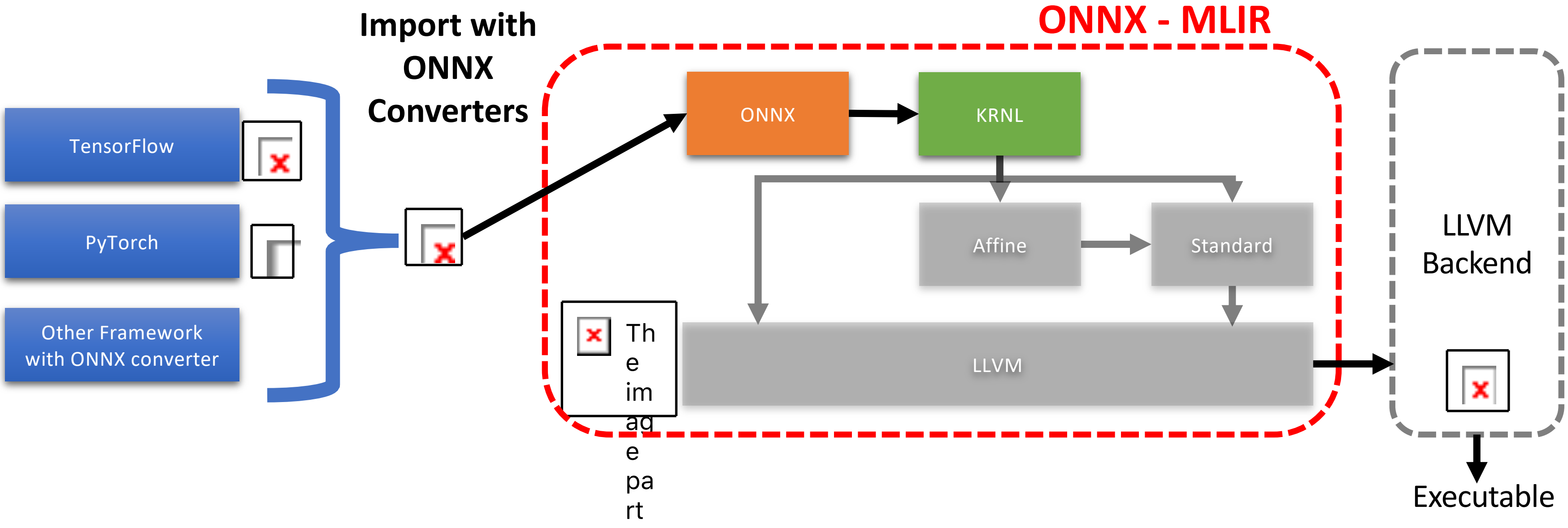
Multi-Level Intermediate Representation (MLIR)

- **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.


TensorFlow (TF) & TensorFlow Runtime (TFRT)
FLANG (LLVM Fortran)
NPComp (Numerical Python Compiler)
ONNX MLIR
dialects
LinAlg
Affine
Standard
The image pa
CPU Optimized Computations
GPU Optimized Computations
Accelerator Optimized Computations
custom backend
CPU binary
GPU binary
accelerator binary

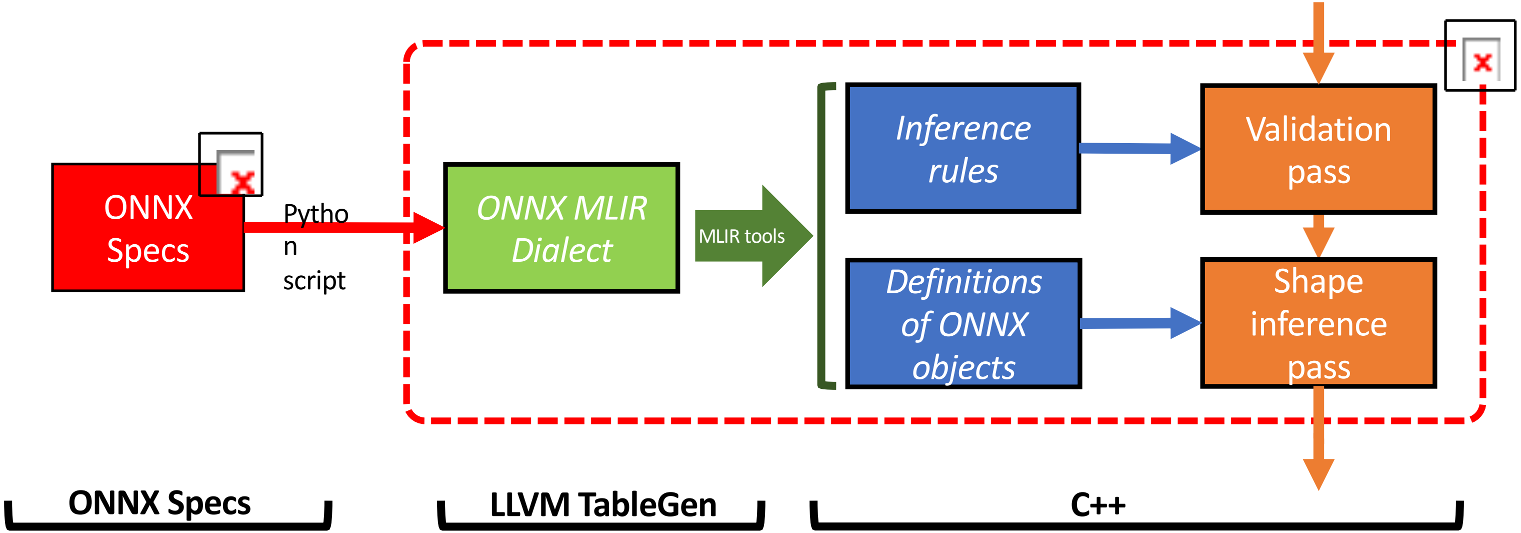
Architecture of ONNX-MLIR Compiler

- **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.

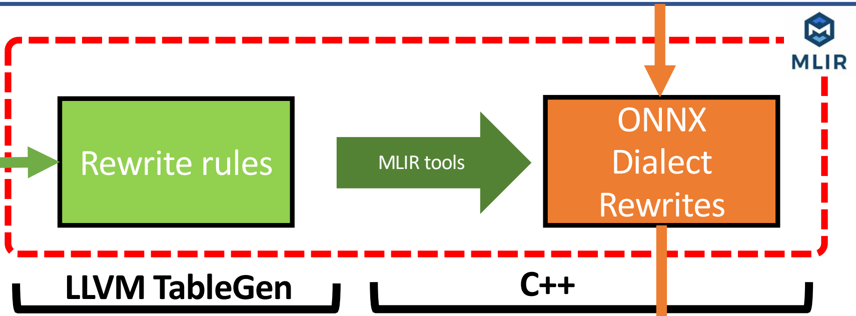


Pattern-Matching Transformations in MLIR

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

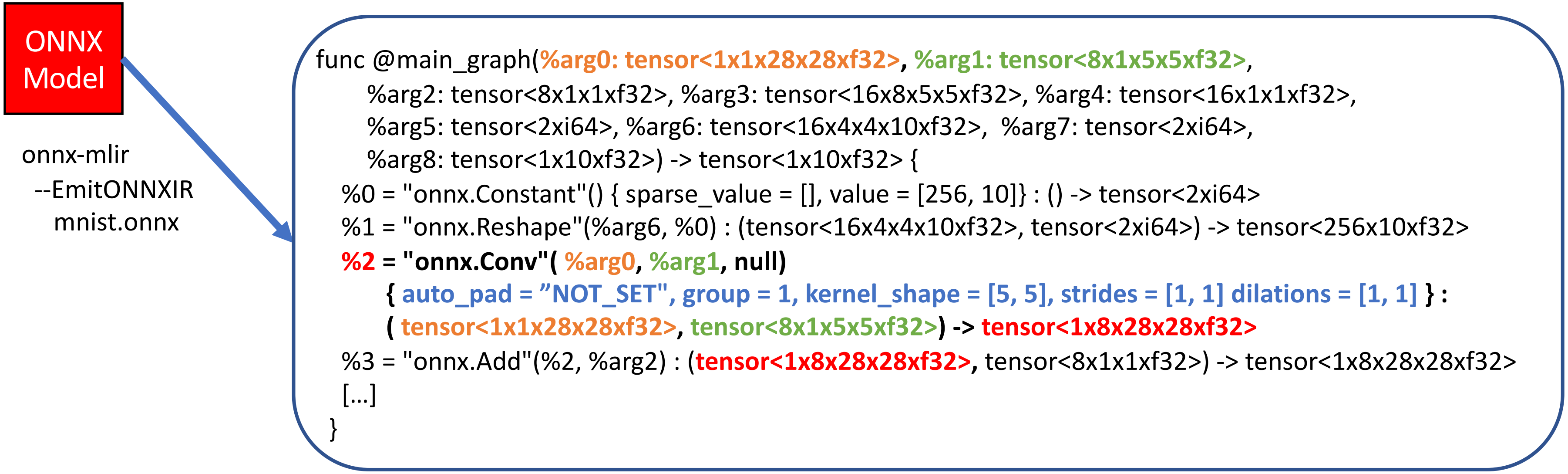
```
%0 = "onnx.MatMul"(%a0, %a1) : (tensor<10x10xf32>, tensor<10x10xf32>) -> tensor<10x10xf32>
%1 = "onnx.Add"(%0, %a2) : (tensor<10x10xf32>, tensor
```

```
Pat<(ONNXAddOp (
        ONNXMatMulOp:$res $m1, $m2),
        $m3),
    (ONNXGemmOp $m1, $m2, $m3,..) >
```



```
%0 = "onnx.Gemm"(%a0, %a1, %a2)
    { alpha = 1.000000e+00, beta = 1.000000e+00,  transA = 0, transB = 0} :
    ( tensor<10x10xf32>, tensor<10x10xf32>, tensor<10x10xf32>) -> tensor<10x10xf32>
```

Example: MNIST model to ONNX dialect



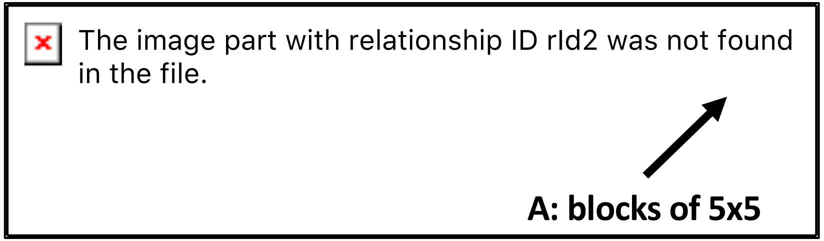
onnx-mlir
--EmitONNXIR
mnist.onnx

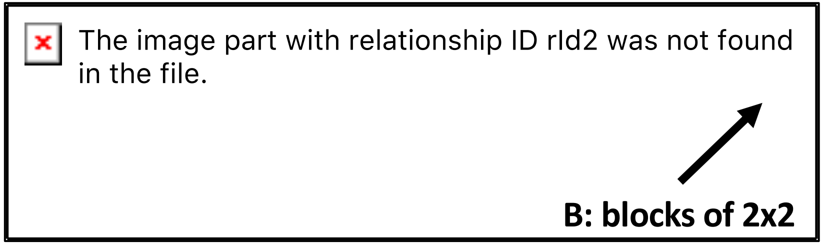
```
func @main_graph(%arg0: tensor<1x1x28x28xf32>, %arg1: tensor<8x1x5x5xf32>,
      %arg2: tensor<8x1x1xf32>, %arg3: tensor<16x8x5x5xf32>, %arg4: tensor<16x1x1xf32>,
      %arg5: tensor<2xi64>, %arg6: tensor<16x4x4x10xf32>,  %arg7: tensor<2xi64>,
      %arg8: tensor<1x10xf32>) -> tensor<1x10xf32> {
   %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.

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:

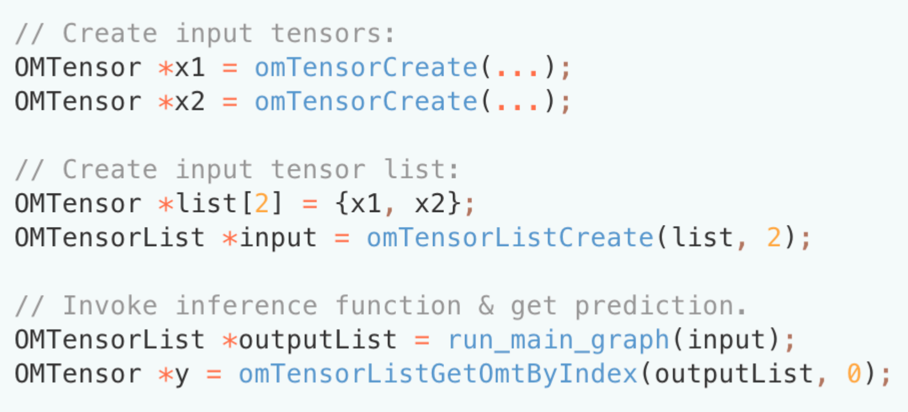

The image part with relationship ID rId2 was not found in the file.


The image part with relationship ID rId2 was not found in the file.

- 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!

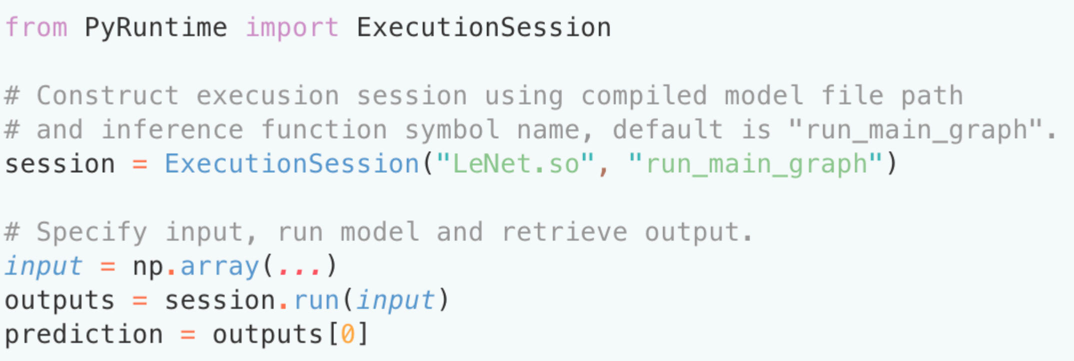
ONNX-MLIR Runtime Interface

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



```
// 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 = omTensorListGetOmtByIndex(outputList, 0);
```

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

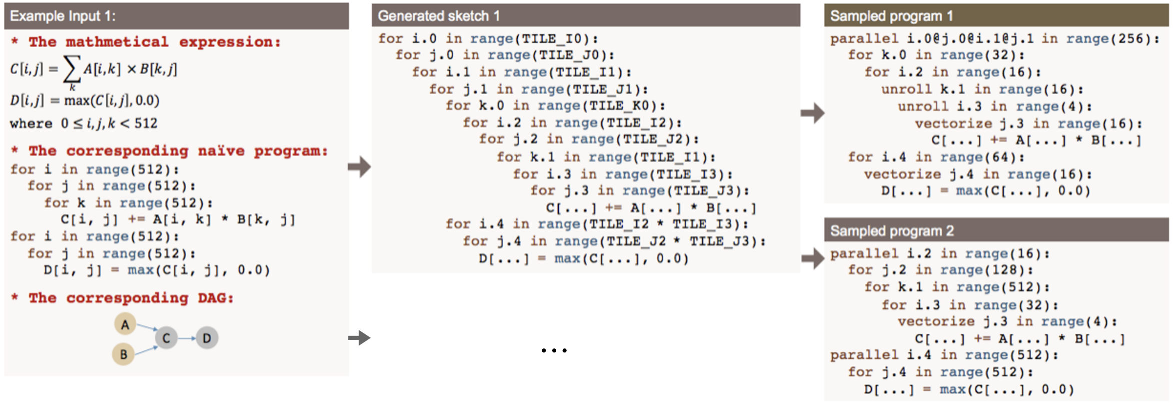


```
from PyRuntime import ExecutionSession

# Construct execusion 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**
  - Additional documentation: **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



Example Input 1:

* The mathmetical expression:

$C[i,j] = \sum_k A[i,k] \times B[k,j]$

$D[i,j] = \max(C[i,j], 0.0)$

where $0 \le i,j,k < 512$

* The corresponding naïve program:

```
for i in range(512):
  for j in range(512):
    for k in range(512):
      C[i, j] += A[i, k] * B[k, j]
for i in range(512):
  for j in range(512):
    D[i, j] = max(C[i, j], 0.0)
```

* The corresponding DAG:

Generated sketch 1

```
for i.0 in range(TILE_I0):
  for j.0 in range(TILE_J0):
    for i.1 in range(TILE_I1):
      for j.1 in range(TILE_J1):
        for k.0 in range(TILE_K0):
          for i.2 in range(TILE_I2):
            for j.2 in range(TILE_J2):
              for k.1 in range(TILE_I1):
                for i.3 in range(TILE_I3):
                  for j.3 in range(TILE_J3):
                    C[...] += A[...] * B[...]
        for i.4 in range(TILE_I2 * TILE_I3):
          for j.4 in range(TILE_J2 * TILE_J3):
            D[...] = max(C[...], 0.0)
```

...

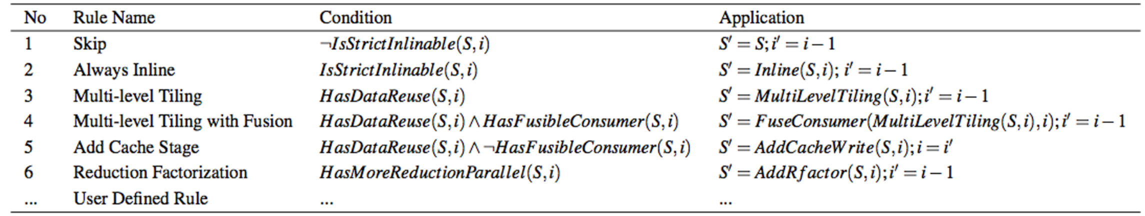
Sampled program 1

```
parallel i.0@j.0@i.1@j.1 in range(256):
  for k.0 in range(32):
    for i.2 in range(16):
      unroll k.1 in range(16):
        unroll i.3 in range(4):
          vectorize j.3 in range(16):
            C[...] += A[...] * B[...]
  for i.4 in range(64):
    vectorize j.4 in range(16):
      D[...] = max(C[...], 0.0)
```

Sampled program 2

```
parallel i.2 in range(16):
  for j.2 in range(128):
    for k.1 in range(512):
      for i.3 in range(32):
        vectorize j.3 in range(4):
          C[...] += A[...] * B[...]
parallel i.4 in range(512):
  for j.4 in range(512):
    D[...] = max(C[...], 0.0)
```

Widens search space even further than AutoTVM 1.0



No	Rule Name	Condition	Application
1	Skip	$\neg IsStrictInlinable(S,i)$	$S' = S; i' = i-1$
2	Always Inline	$IsStrictInlinable(S,i)$	$S' = Inline(S,i); i' = i-1$
3	Multi-level Tiling	$HasDataReuse(S,i)$	$S' = MultiLevelTiling(S,i); i' = i-1$
4	Multi-level Tiling with Fusion	$HasDataReuse(S,i) \wedge HasFusibleConsumer(S,i)$	$S' = FuseConsumer(MultiLevelTiling(S,i),i); i' = i-1$
5	Add Cache Stage	$HasDataReuse(S,i) \wedge \neg HasFusibleConsumer(S,i)$	$S' = AddCacheWrite(S,i); i = i'$
6	Reduction Factorization	$HasMoreReductionParallel(S,i)$	$S' = AddRfactor(S,i); i' = i-1$
...	User Defined Rule	...	...

Table 1: Derivation rules used to generate sketches. The condition runs on the current state $\sigma = (S,i)$. The application derives the next state $\sigma' = (S',i')$ from the current state σ. Note that some function (*e.g.*, $AddRfactor$, $FuseConsumer$) can return multiple possible values of S'. In this case we collect all possible S', and return multiple next states σ' for a single input state σ.

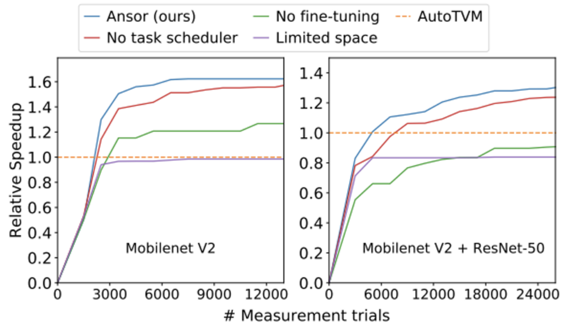


Figure 10: Network performance auto-tuning curve. The y-axis is the speedup relative to AutoTVM.

Ansor: Generating High-Performance Tensor Programs for Deep Learning Zheng L, et al. 2020 https://arxiv.org/pdf/2006.06762.pdf

15 Slides/9 minutes

OctoML’s wishlist for ONNX

We wish ONNX had…

- Even broader op coverage (eg `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!

Microsoft

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)

ONNX RUNTIME TensorRT tvm

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

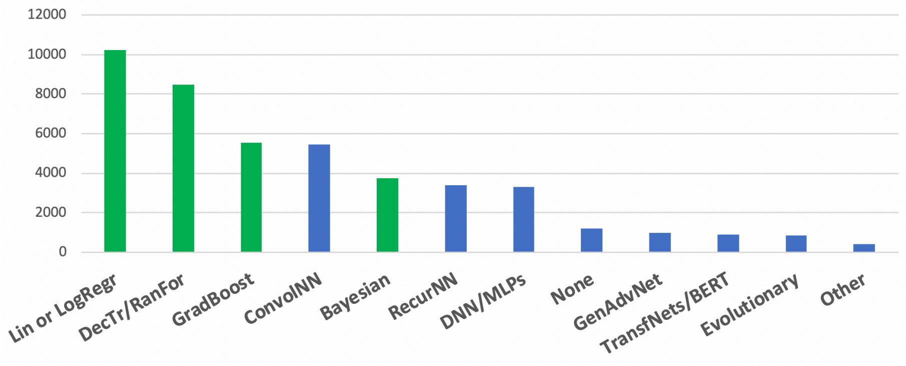
scikit learn ML.NET H_2O.ai

Traditional ML Models

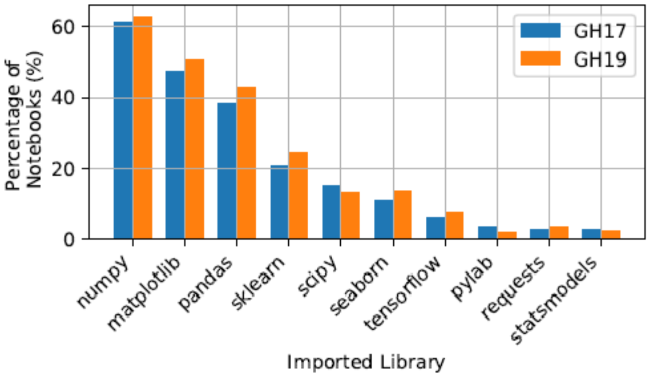
ML.NET

scikit learn

H_2O.ai



2019 Kaggle Survey: The State of Data Science & Machine Learning



Data Science throw the looking glass: https://arxiv.org/abs/1912.09536

Hummingbird

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

Benefits:

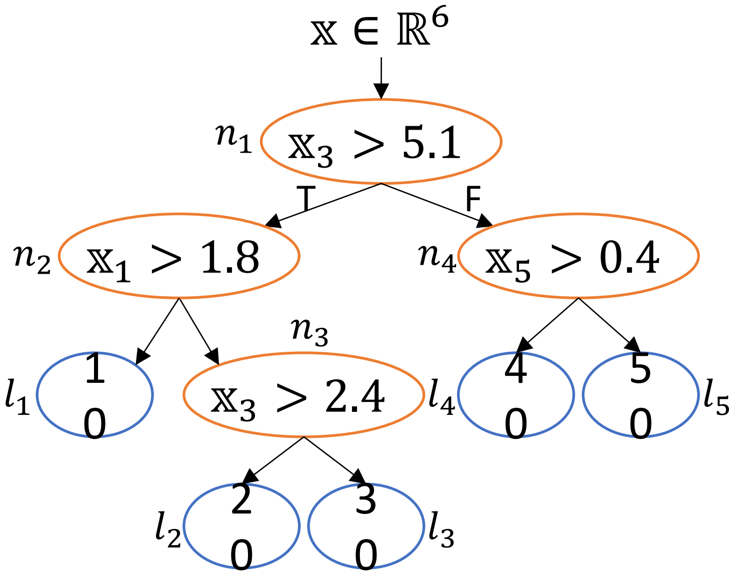
(1) Exploit the already available DNN runtimes
(2) Exploit current (and future DNN) optimizations
(3) Seamless hardware acceleration
(4) Significant reduction in engineering effort

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!

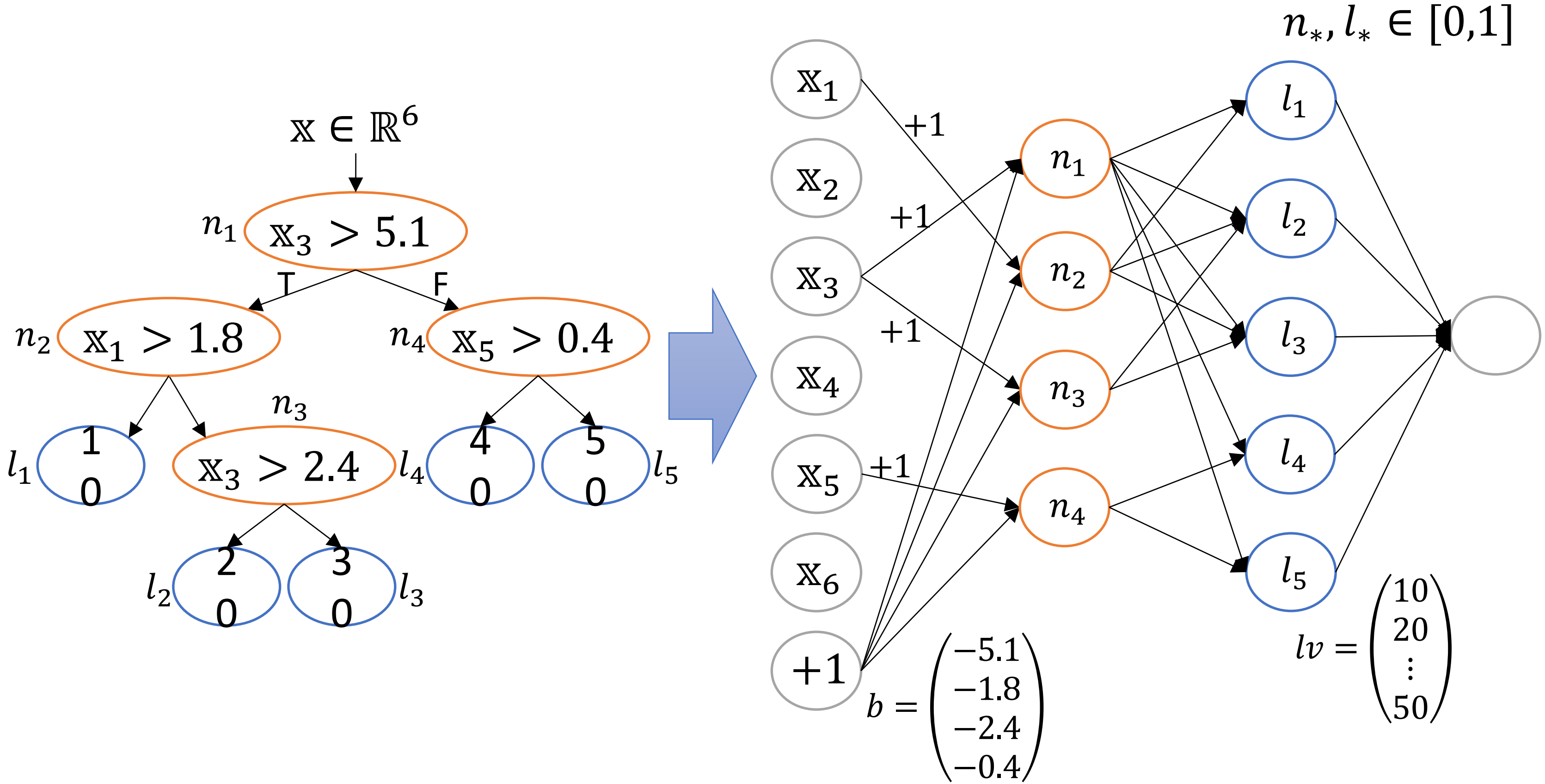
Translating Trees



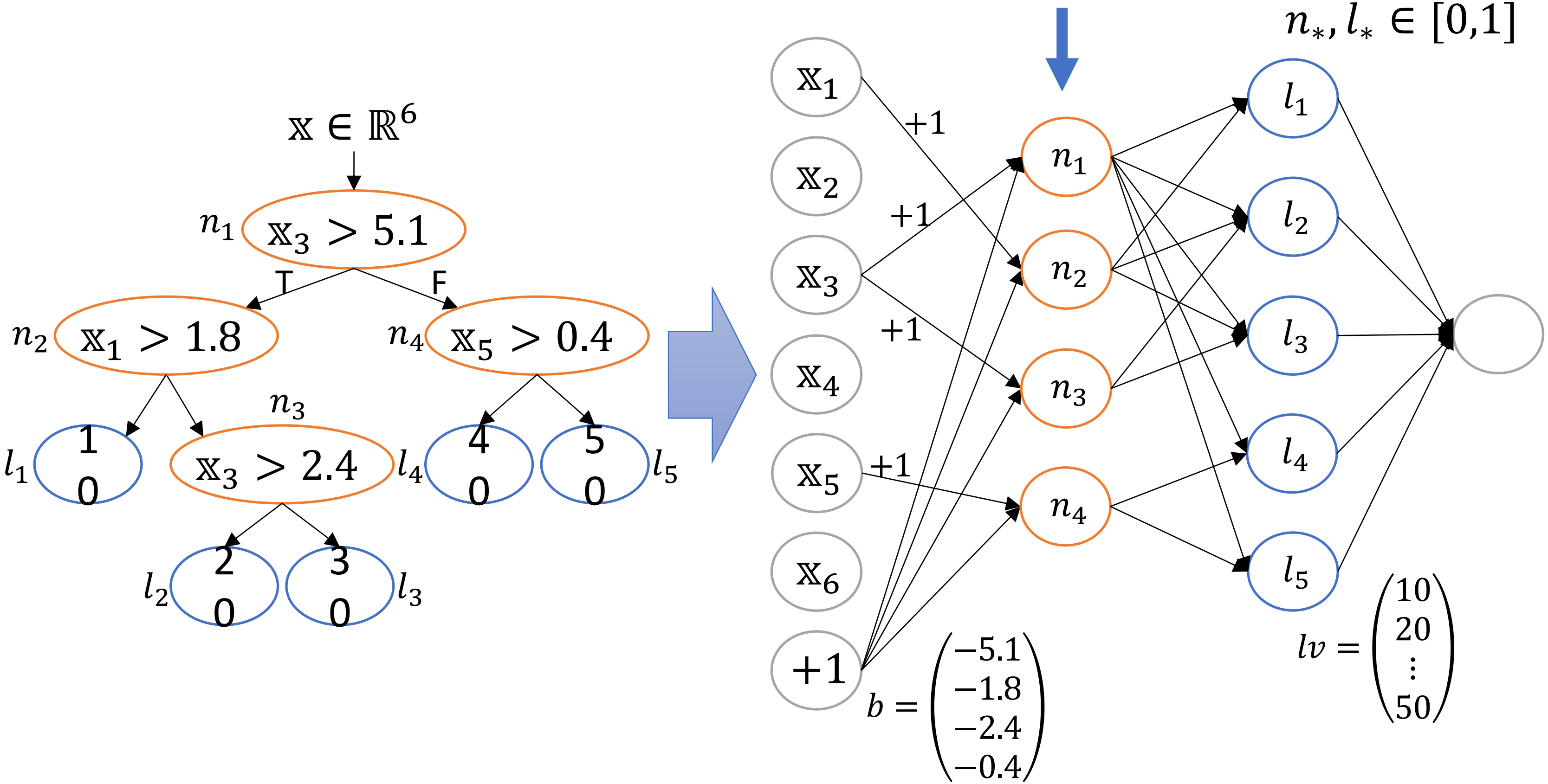
Internal node: n_*

Leaf node: l_*

Translating Trees



Translating Trees



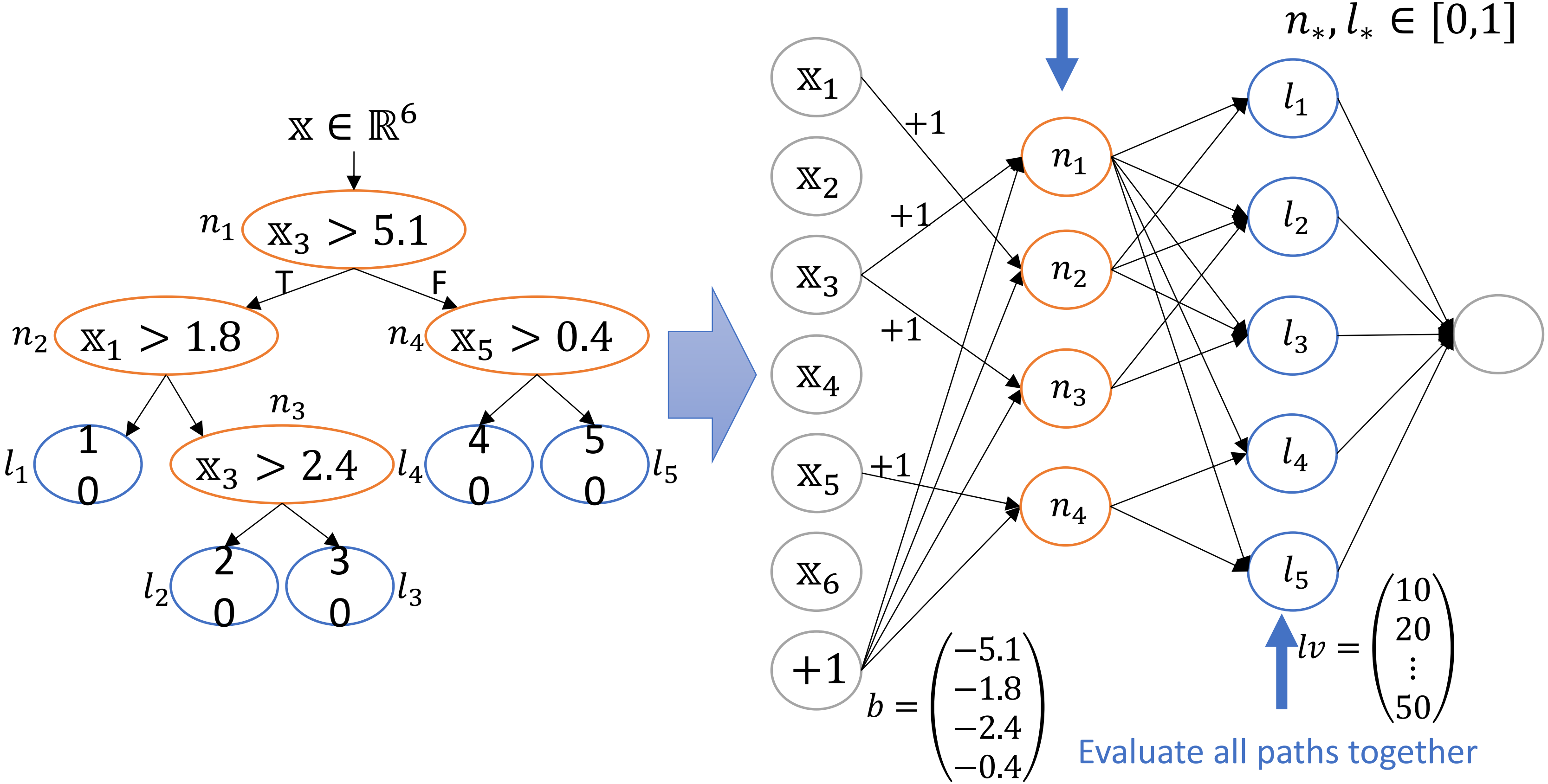
Evaluate all conditions together

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

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

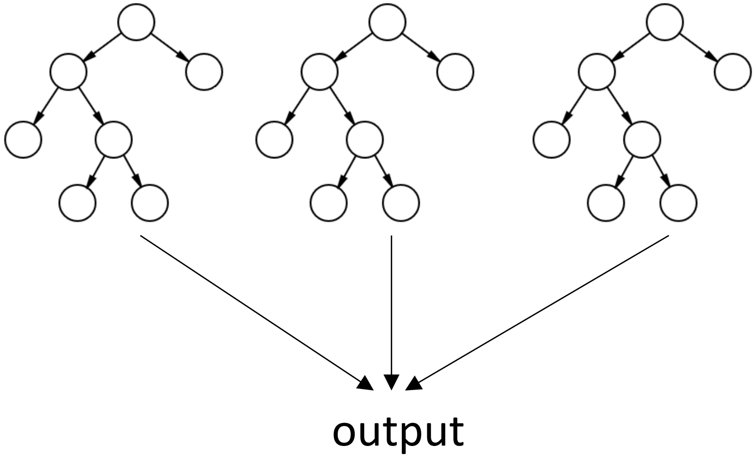
$$lv = \begin{pmatrix} 10 \\ 20 \\ \vdots \\ 50 \end{pmatrix}$$

Translating Trees

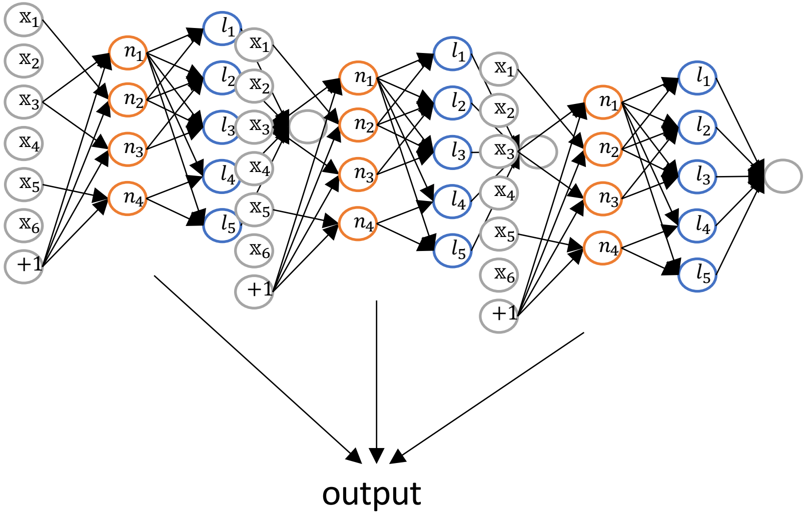


Translating Trees

- Random forest, boosting, …



output

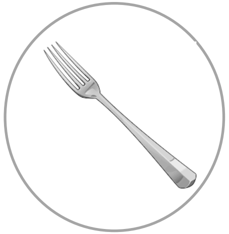


output

Hummingbird: Status

- Open sourced in May: 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

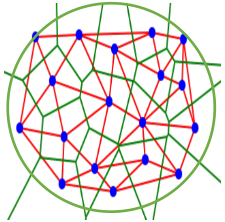
1.7K GitHub stars



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



Used by 9

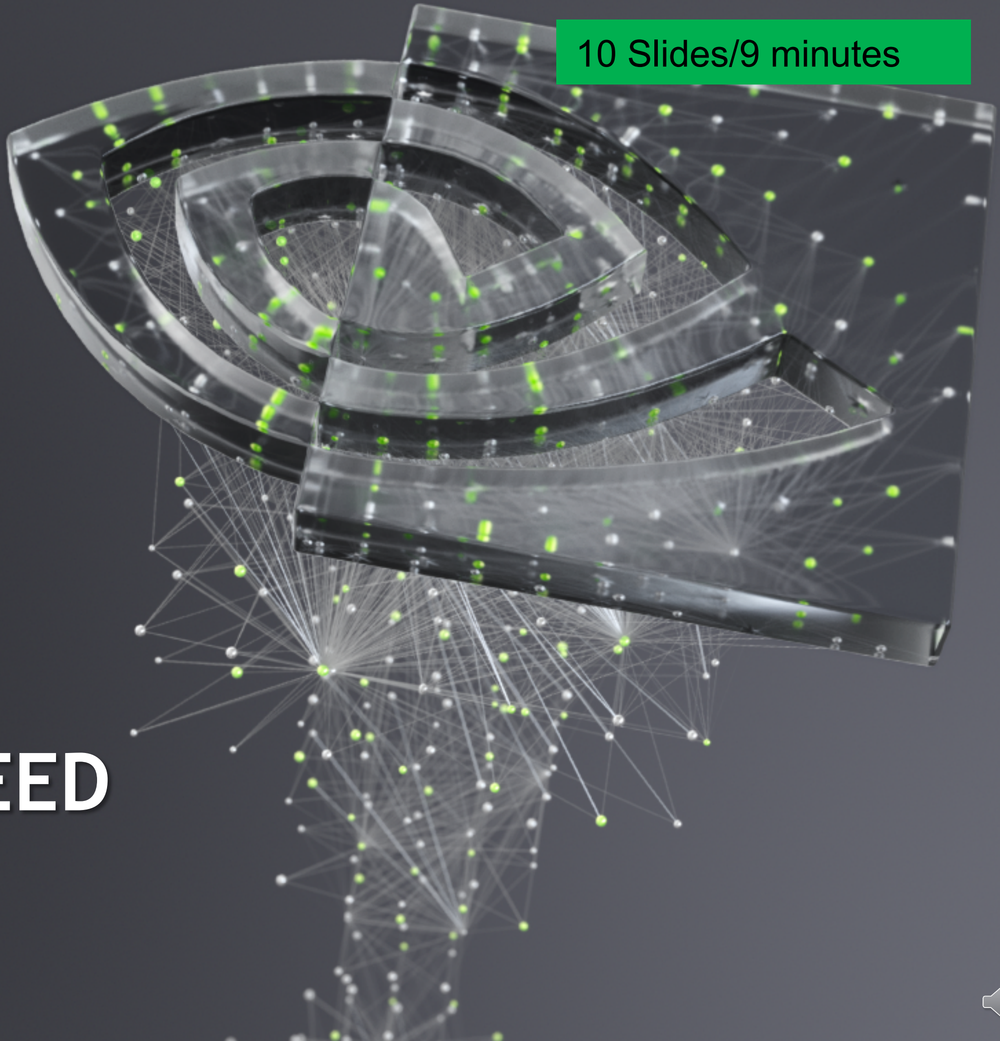
Future work: Integration with other ONNX converters

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

Microsoft

Thank you!

hummingbird-dev@microsoft.com



10 Slides/9 minutes

Q/DQ IS ALL YOU NEED

Neta Zmora,

Oct 14, 2020

AGENDA

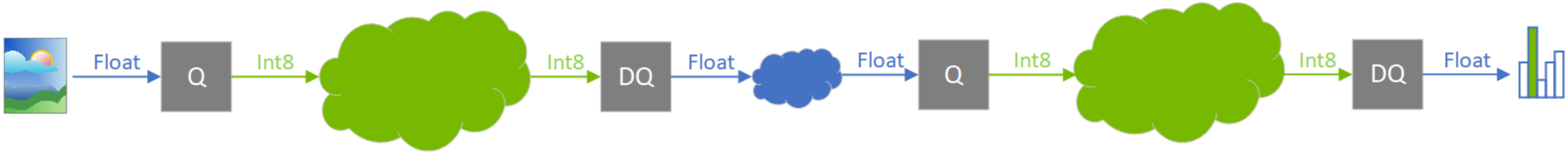
Q/DQ are necessary and sufficient

How do we optimize graphs with Q/DQ?

Where do we insert Q/DQ in our graphs?

Q/DQ ARE NECESSARY

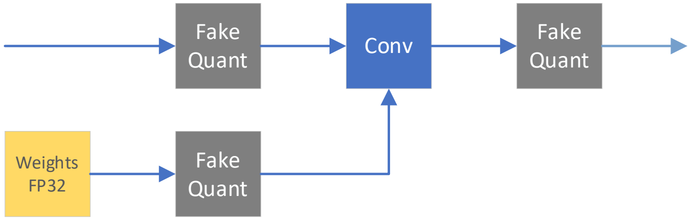
- For quantizing network input; and dequantizing network output.
- For changing precision mid-graph (for example: input to softmax).



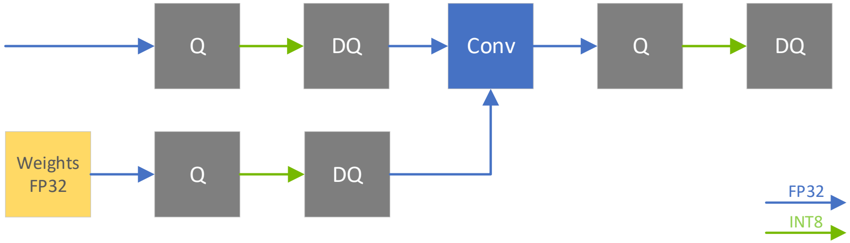
QAT, PTQ AND FAKE QUANTIZATION

- Fake quantization is the prevailing approach used for DNN quantization.
- Forward pass: $\hat{x} = \mathrm{dequantize}(\mathrm{quantize}(x))$
- ONNX QuantizeLinear and DequantizeLinear naturally represent fake quantization.

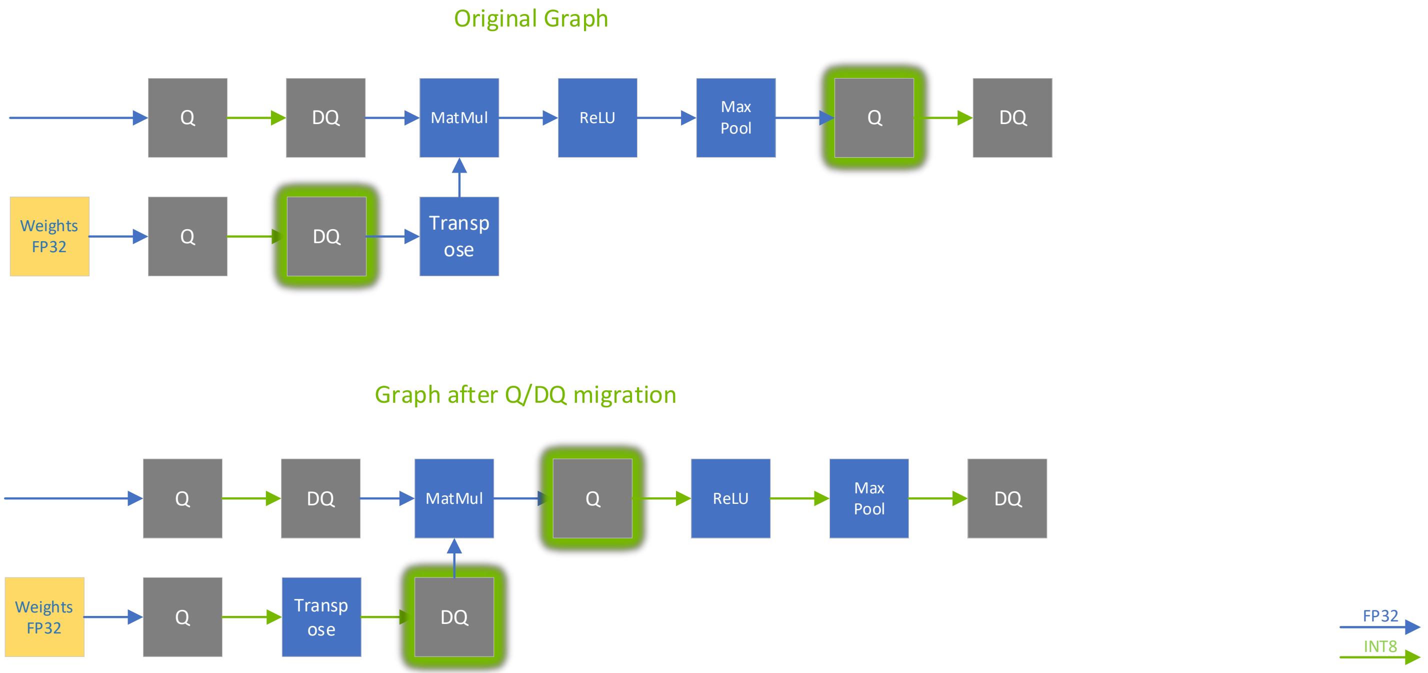
Fake Quantization (QDQ) in training framework



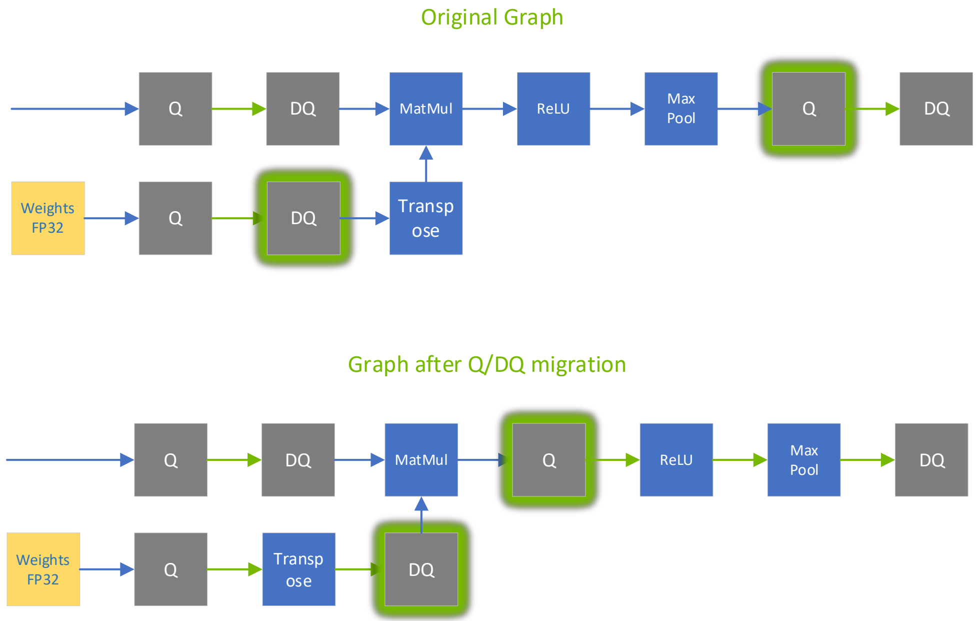
Fake Quantization (QDQ) in ONNX



Minimize activation bandwidth



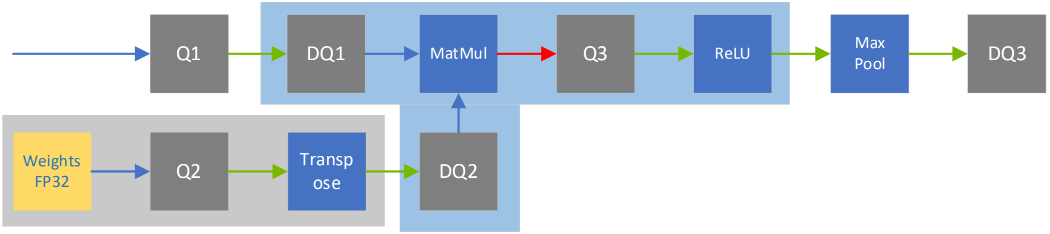
*In this presentation we assume per-tensor quantization for activations, and per-channel quantization for weights. This produces the best results and simplifies the math.



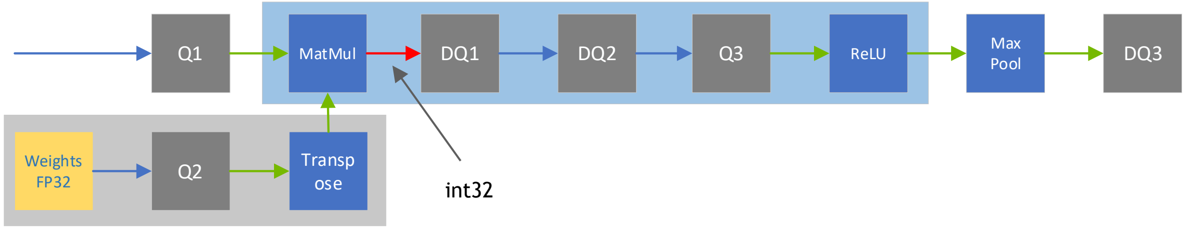
Commuting with Scale/Shift:

- OP -> Q =? Q -> OP
- DQ -> OP =? OP -> DQ

Fusion Opportunities



MatMul Fusion Reordering



CHANGING Q/DQ OPERATION ORDER

Left as an offline exercise for the reader

Consider a scalar multiplication:

*c = a*b*

Suppose fake quantization nodes are attached to the input:

$$\begin{aligned} c &= \hat{c} \\ \hat{c} &= qdq(a)qdq(b) \\ &= ((a_q - z_a)s_a)((b_q - z_q)s_b) \\ &= s_a s_b(a_q b_q - (z_b a_q + z_a b_q - z_a z_b)) \end{aligned}$$

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

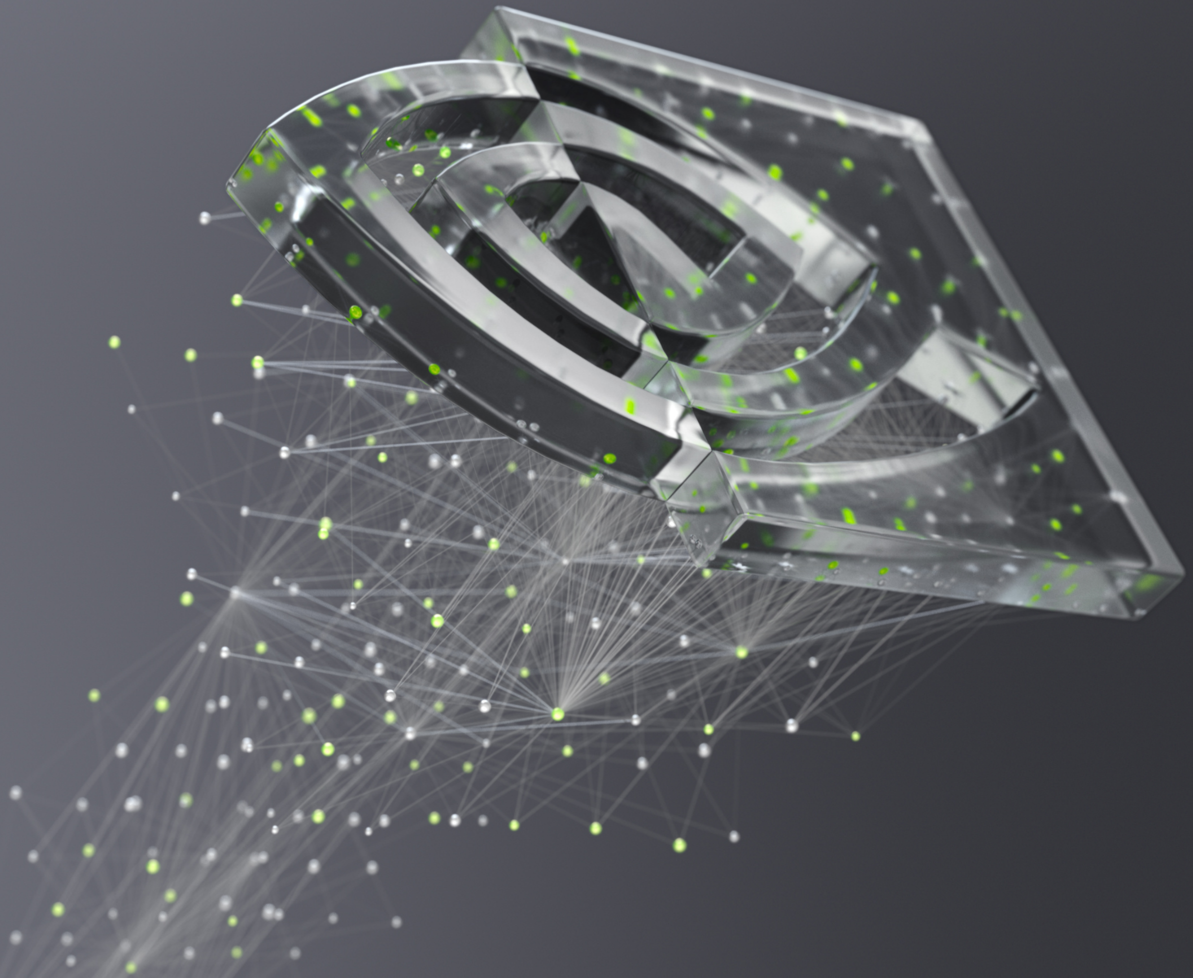
$$s_c = s_a s_b$$

$$z_c = z_b a_q + z_a b_q - z_a z_b$$

➔ 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.



ONNX

Wish we had more time!

ON-BOARDING ONNX MODELS IN ACUMOS

(BY @PHILIPPE LFAI/ONXX SLACK)

Philippe Dooze
Orange

- **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.

ONNX

SIG Presentations

4 Slides/9 minutes

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
- 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 version3)
- 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

14 Slides/9 minutes

ONNX

OPERATORS SIG

BREAKOUT SESSION

Emad Barsoum (Microsoft)
Michal Karzynski (Intel)

AGENDA

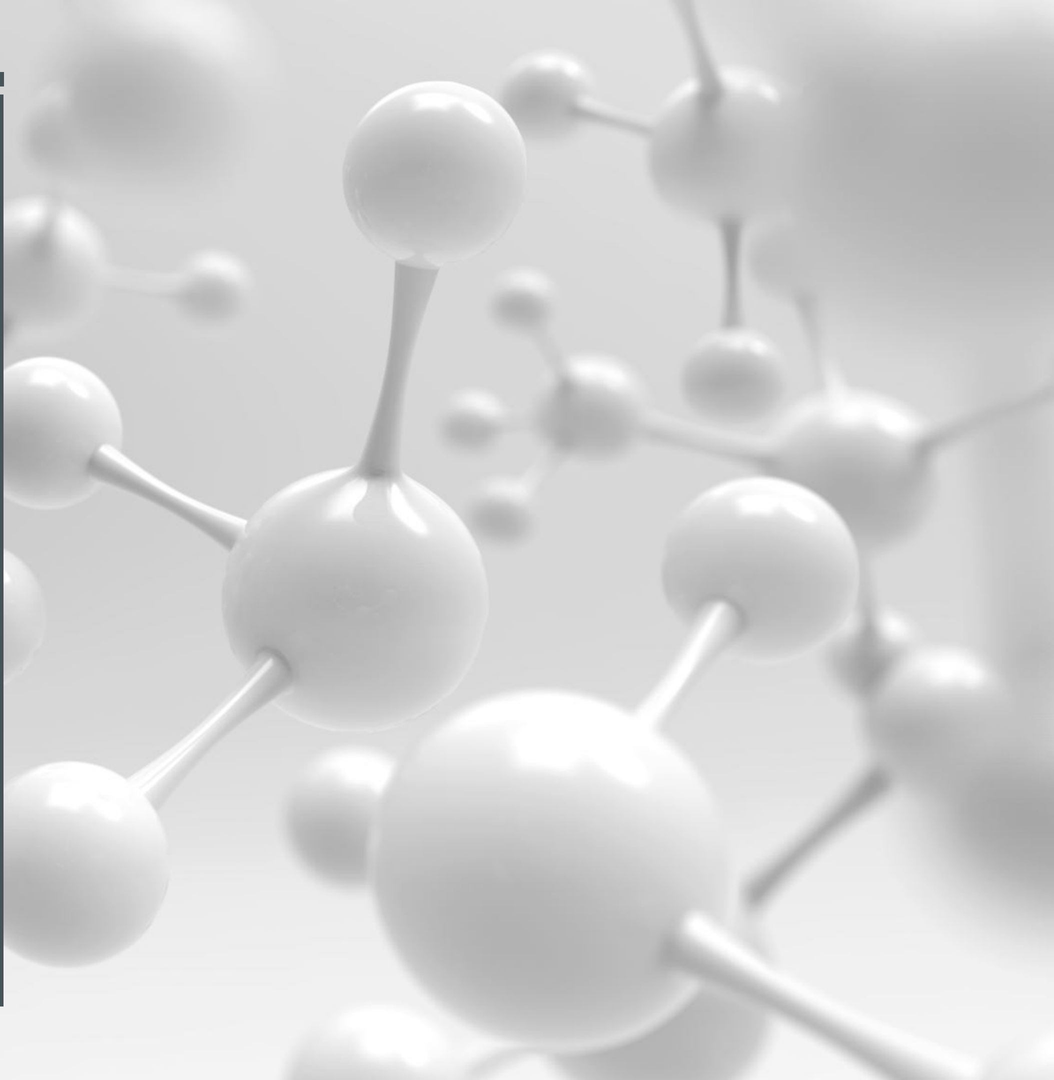
- Operators SIG
- Add new operator update
- Proposal / improvement
- Discussion:Version converter
- Discussion: PR and Issues

GOAL

Keep Up	Quality	Clarity	Size	PRs and Issues
Keep up with the latest progress in AI	Improve the quality of ONNX Operators	Reduce ambiguity	Avoid bloating ONNX spec	Keep up with PRs and operator issues

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 Oljaca (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://gitter.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.

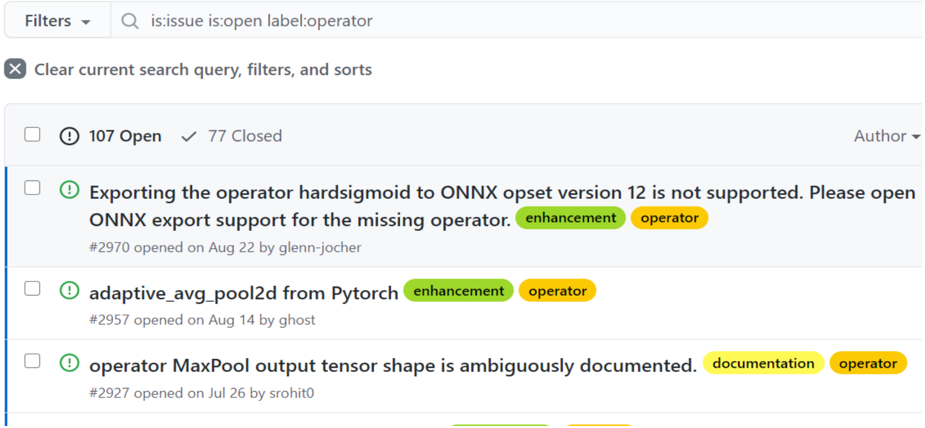
ADDING NEW OPERATOR UPDATE

- 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



Filters ▾ is:issue is:open label:operator

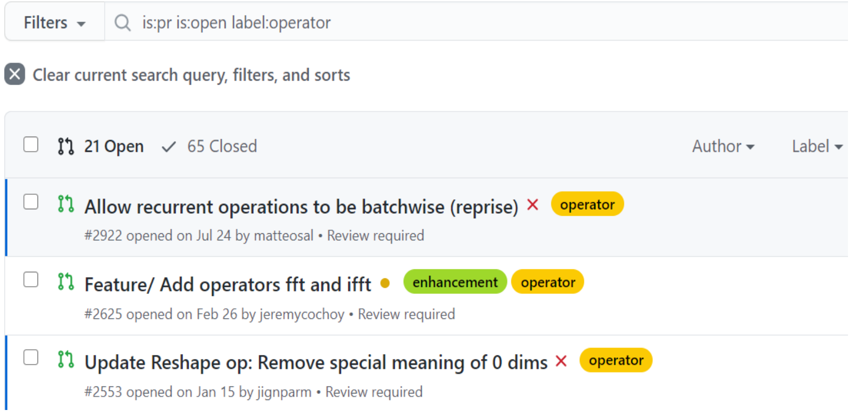
☒ Clear current search query, filters, and sorts

□ 107 Open ✓ 77 Closed Author ▾

□ Exporting the operator hardsigmoid to ONNX opset version 12 is not supported. Please open ONNX export support for the missing operator. enhancement operator
#2970 opened on Aug 22 by glenn-jocher

□ adaptive_avg_pool2d from Pytorch enhancement operator
#2957 opened on Aug 14 by ghost

□ operator MaxPool output tensor shape is ambiguously documented. documentation operator
#2927 opened on Jul 26 by srohit0



Filters ▾ is:pr is:open label:operator

☒ Clear current search query, filters, and sorts

□ 21 Open ✓ 65 Closed Author ▾ Label ▾

□ Allow recurrent operations to be batchwise (reprise) ✗ operator
#2922 opened on Jul 24 by matteosal • Review required

□ Feature/ Add operators fft and ifft ● enhancement operator
#2625 opened on Feb 26 by jeremycochoy • Review required

□ Update Reshape op: Remove special meaning of 0 dims ✗ operator
#2553 opened on Jan 15 by jignparm • Review required

PR REVIEW

- PRs should be marked with the Operator label
  - 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 and join onnx-operators
- Documents and artifacts: https://github.com/onnx/sigs/tree/master/operators

11 Slides/9 minutes

Converters SIG Updates

ONNX Workshop 10/14/2020

Chin Huang, IBM

Guenther Schmuelling, Microsoft

Kevin Chen, Nvidia

Converters SIG Updates

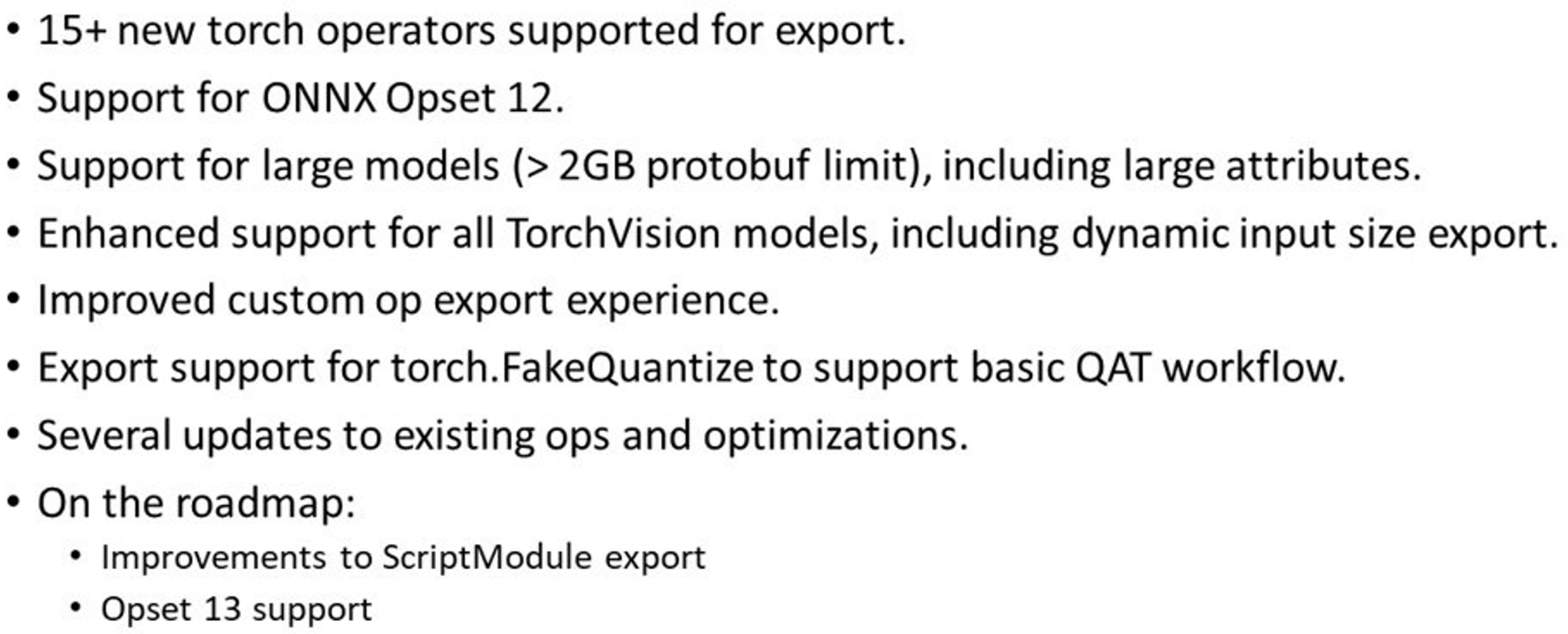
- Converters updates
  - Frontend converters
  - Backend converters

- 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/

8 Slides/9 minutes

ONNX Model Zoo + Tutorials SIG Update

Wenbing Li, Microsoft
Vinitra Swamy, EPFL
10/14/2020

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

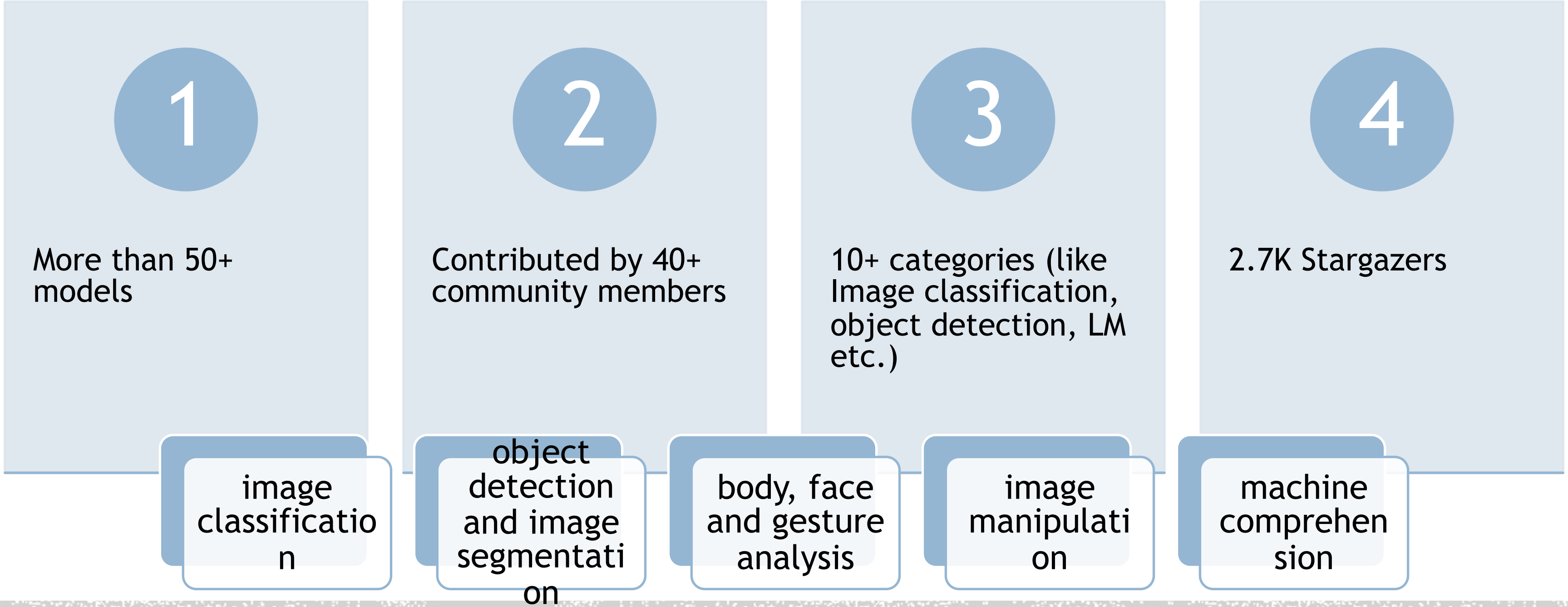
Git LFS Migration complete! (onnx#271)

- 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

ONNX Model Zoo

Key numbers



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

image classification

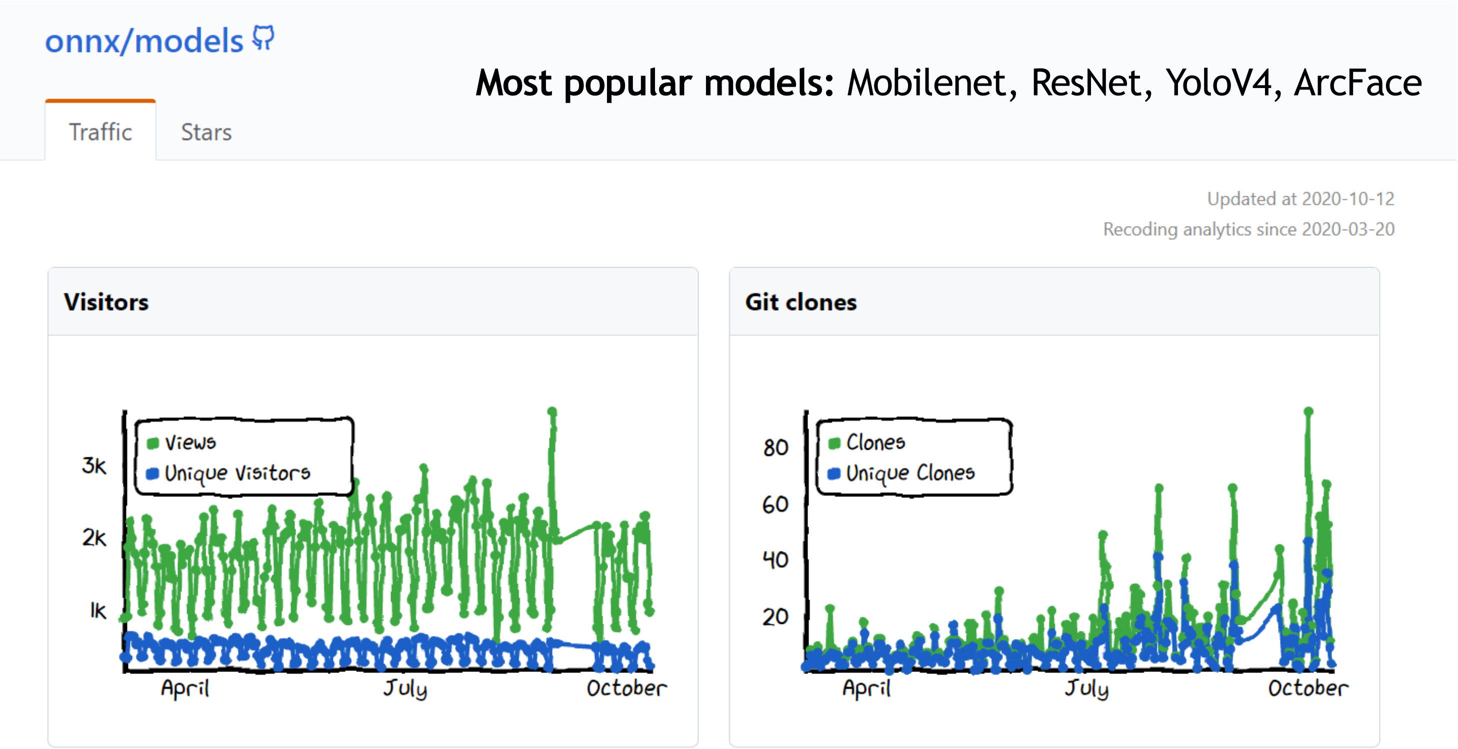
object detection and image segmentation

body, face and gesture analysis

image manipulation

machine comprehension

Most popular models: Mobilenet, ResNet, YoloV4, ArcFace



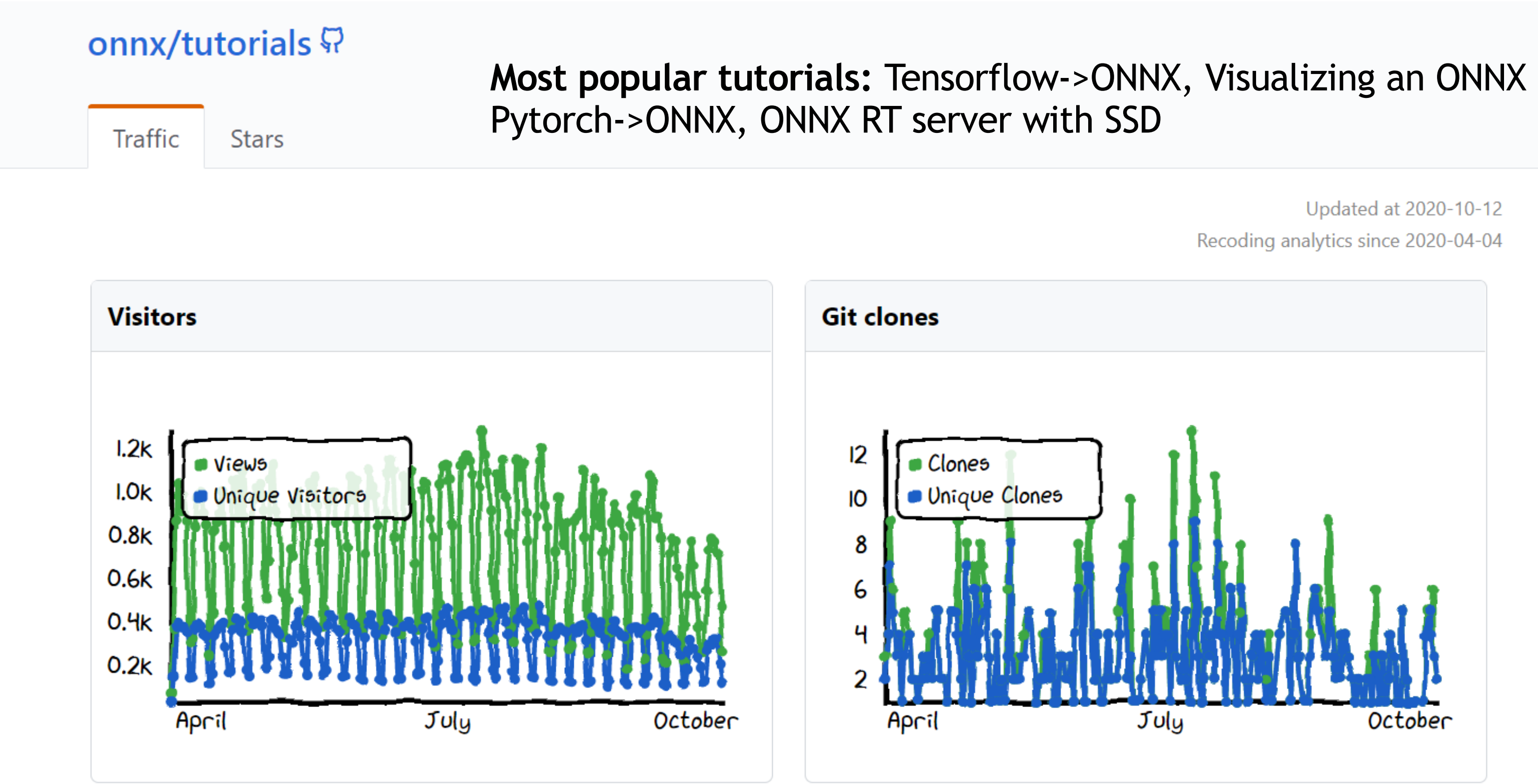
onnx/models

Updated at 2020-10-12
Recoding analytics since 2020-03-20

Visitors

Git clones

Most popular tutorials: Tensorflow->ONNX, Visualizing an ONNX model, Pytorch->ONNX, ONNX RT server with SSD



onnx/tutorials

Traffic

Stars

Updated at 2020-10-12

Recoding analytics since 2020-04-04

Visitors

Git clones

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

Contribute your models

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

Model Zoo SIG Resources

- 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