ONNX Github Repositories

Chin Huang, Cognitive OpenTech

Data and AI
Open Source Dojo

Outline

- Github Repository Categories
  - ONNX Core
  - Converters
  - Getting Started
  - Utility
  - Community
  - ONNX Runtime

ONNX Github Repository Categories

Category	Name	Link
Core	ONNX specs and format	https://github.com/onnx/onnx
Converters	Keras to ONNX	https://github.com/onnx/keras-onnx
	Tensorflow to ONNX	https://github.com/onnx/tensorflow-onnx
	A few ML Toolkits to ONNX	https://github.com/onnx/onnxmltools
	Scikit Learn to ONNX	https://github.com/onnx/sklearn-onnx
	ONNX to Tensorflow	https://github.com/onnx/onnx-tensorflow
	ONNX to TensorRT	https://github.com/onnx/onnx-tensorrt
	ONNX to MLIR	https://github.com/onnx/onnx-mlir
	ONNX to Core ML	https://github.com/onnx/onnx-coreml

ONNX Github Repository Categories

Category	Name	Link
Getting started	Model zoo	https://github.com/onnx/models
	Tutorials	https://github.com/onnx/tutorials
Utility	Wheel builder	https://github.com/onnx/wheel-builder
	Docker build scripts	https://github.com/onnx/onnx-docker
	Backend scoreboard	https://github.com/onnx/backend-scoreboard
Community	Steering committee	https://github.com/onnx/steering-committee
	SIG artifacts	https://github.com/onnx/sigs
	Working groups	https://github.com/onnx/working-groups
	For https://onnx.ai/	https://github.com/onnx/onnx.github.io
Language	R interface to ONNX	https://github.com/onnx/onnx-r

ONNX Core: onnx/onnx

Key components and features in ONNX Core

- Model spec/schema
  - Operator sets: operators, ML operators, training operators
  - Serialization, documentation, APIs
- Tools
  - Shape inference (C++, python)
  - Optimizer (C++, python)
  - Checker (C++, python)
  - ONNX Interface for Framework Integration (C++)
- Unit tests: Dummy backend, other backends
- CI: travis (Linux, Mac OSx), Azure pipelines (Windows)

ONNX Converters

Under ONNX organization

- Frontend: Tensorflow, Keras, Scikit-Learn, MLTools
- Backend: Tensorflow, TensorRT, Core ML, MLIR

Outside of ONNX

- Examples: Chainer, Caffe2, MatLab, PaddlePaddle, and more
- Complete list: https://github.com/onnx/sigs/blob/master/converters/docs/ConvertersList.md

ONNX Model Zoo: onnx/model

- A collection of pre-trained, state-of-the-art models in the ONNX format, https://github.com/onnx/models
- Accompanying each model are Jupyter notebooks for model training and running inference with the trained model. The notebooks can be exported and run as python(.py) files.
- Each model has a model.onnx and test data.
- Every ONNX backend should support running these models out of the box!
- Not all categories supported… Contribution welcome!

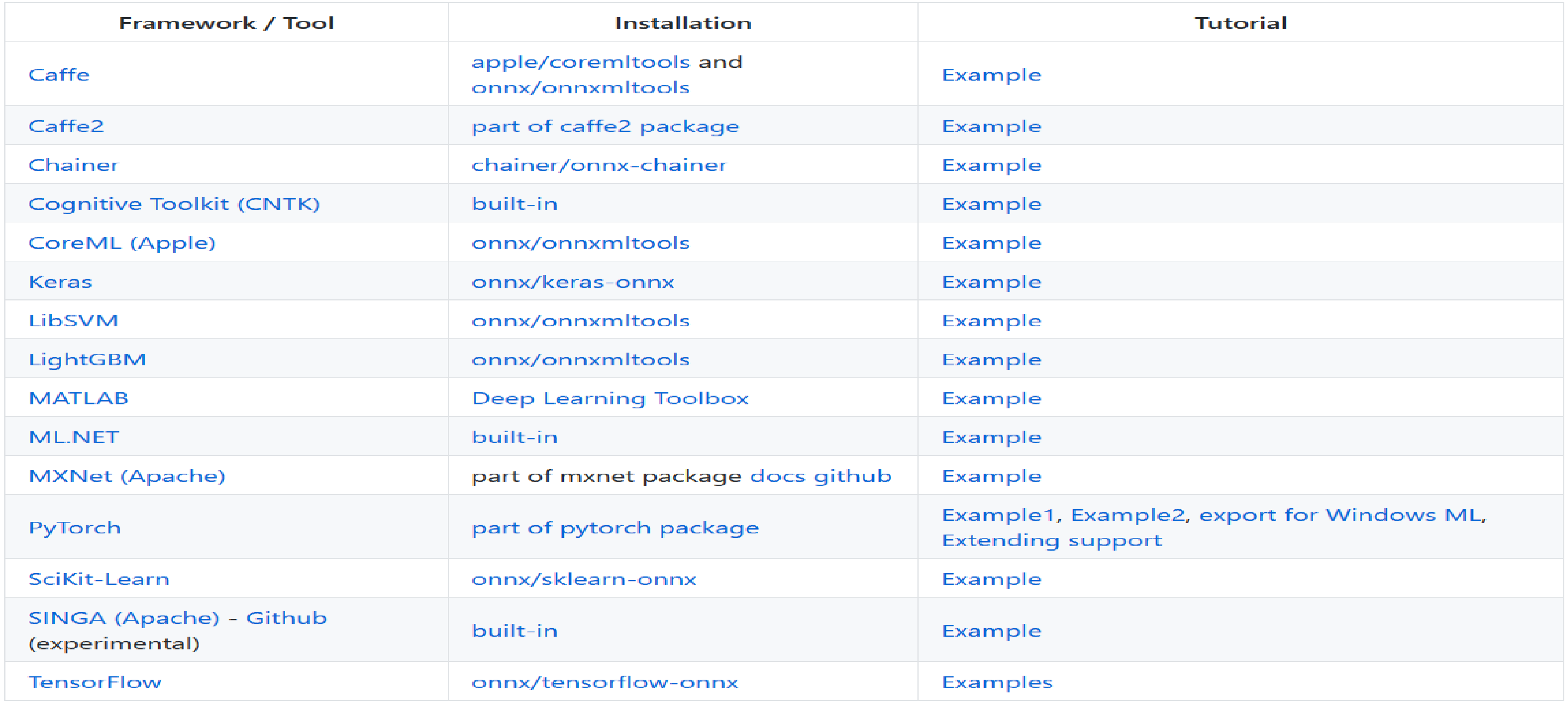
ONNX Model Zoo – Categories and Models

Category	Model	Comments
Image Classification	MobileNet, ResNet, SqueezeNet, VGG, Bvlc_AlexNet, Bvlc_GoogleNet, Bvlc_reference_CaffeNet, Bvlc_reference_RCNN_ILSVRC13, DenseNet121, Inception_v1, Inception_v2, ShuffleNet, ZFNet512	This collection of models take images as input, then classifies the major objects in the images into a set of predefined classes.
Face Detection and Recognition	ArcFace	Detect and/or recognize human faces in images
Semantic Segmentation	DUC	Partition an input image by labeling each pixel into a set of pre-defined categories
Object Detection & Segmentation	Tiny_YOLOv2	Detect the presence of multiple objects in an image and segment out areas of the image
Emotion Recognition	Emotion FerPlus	
Hand Written	MNIST- Hand Written Digit Recognition	

ONNX Tutorials: onnx/tutorials

- Tutorials for using ONNX with different framework, https://github.com/onnx/tutorials
- Terminology
    - Export = convert models from framework to ONNX
    - Import = convert models from ONNX to framework
- Most tutorials are one way conversion.
- Some end-to-end tutorials including two frameworks.

ONNX Tutorials: Export from Frameworks



Framework / Tool	Installation	Tutorial
Caffe	apple/coremltools and onnx/onnxmltools	Example
Caffe2	part of caffe2 package	Example
Chainer	chainer/onnx-chainer	Example
Cognitive Toolkit (CNTK)	built-in	Example
CoreML (Apple)	onnx/onnxmltools	Example
Keras	onnx/keras-onnx	Example
LibSVM	onnx/onnxmltools	Example
LightGBM	onnx/onnxmltools	Example
MATLAB	Deep Learning Toolbox	Example
ML.NET	built-in	Example
MXNet (Apache)	part of mxnet package docs github	Example
PyTorch	part of pytorch package	Example1, Example2, export for Windows ML, Extending support
SciKit-Learn	onnx/sklearn-onnx	Example
SINGA (Apache) - Github (experimental)	built-in	Example
TensorFlow	onnx/tensorflow-onnx	Examples

ONNX Tutorials: Scoring ONNX Models

Framework / Tool	Installation	Tutorial
Caffe2	Caffe2	Example
Cognitive Toolkit (CNTK)	built-in	Example
CoreML (Apple)	onnx/onnx-coreml	Example
MATLAB	Deep Learning Toolbox Converter	Documentation and Examples
Menoh	Github Packages or from Nuget	Example
ML.NET	Microsoft.ML Nuget Package	Example
MXNet (Apache) - Github	MXNet	API Example
ONNX Runtime	Python (Pypi) - onnxruntime and onnxruntime-gpu C/C# (Nuget) - Microsoft.ML.OnnxRuntime and Microsoft.ML.OnnxRuntime.Gpu	APIs: Python, C#, C, C++ Examples - Python, C#, C
SINGA (Apache) - Github [experimental]	built-in	Example
Tensorflow	onnx-tensorflow	Example
TensorRT	onnx-tensorrt	Example
Windows ML	Pre-installed on Windows 10	API Tutorials - C++ Desktop App, C# UWP App Examples

ONNX Tutorials: End-to-end Tutorials

- Conversion to deployment: to mobile device (iPhone and Android), ONNX Runtime, Azure ML, AWS
- Serving: with MXNet, ONNX Runtime, Azure ML, AWS SageMaker
- ONNX as an intermediary format
- ONNX Custom Operators

ONNX Wheel Builder: onnx/wheel-builder

- Enables automation of wheel building and deployment of ONNX packages.
- Leverages CI environments
  - Travis
  - AppVeyor (moving to Azure Pipelines soon)
- Publishing destinations
  - TestPypi for release candidates
  - Pypi for official releases
- Dependent repos
  - Multibuild utilities
  - ONNX

ONNX Docker: onnx/onnx-docker

- Stores the docker build scripts of ONNX related docker images.
  - onnx-base: Use published ONNX package from PyPi with minimal dependencies.
  - onnx-dev: Build ONNX from source with minimal dependencies.
  - onnx-ecosystem: Jupyter notebook environment for getting started quickly with ONNX models, ONNX converters, and inference using ONNX Runtime.
- Simple workflow
  - Obtain the Docker images
    - clone this repository and build desired image
    - pull a pre-built image from DockerHub
      - docker pull onnx/onnx-base
      - docker pull onnx/onnx-dev
      - docker pull onnx/onnx-ecosystem
  - Run the images

ONNX Backend Scoreboard: onnx/backend-scoreboard

- Measures the compliance of ONNX backends with the standard ONNX tests.
- Pluggable backends
    - A new backend can be added to the scoreboard following these instructions.
    - The latest stable release is used for ONNX and backend
- Scoring criteria
    - Pass or fail/not supported on the standard ONNX backend tests
    - Essentially the scores represent the coverage of the test cases.
- Participating backends, ONNX-Runtime, ONNX-Tensorflow, nGraph

ONNX Runtime: https://github.com/microsoft/onnxruntime

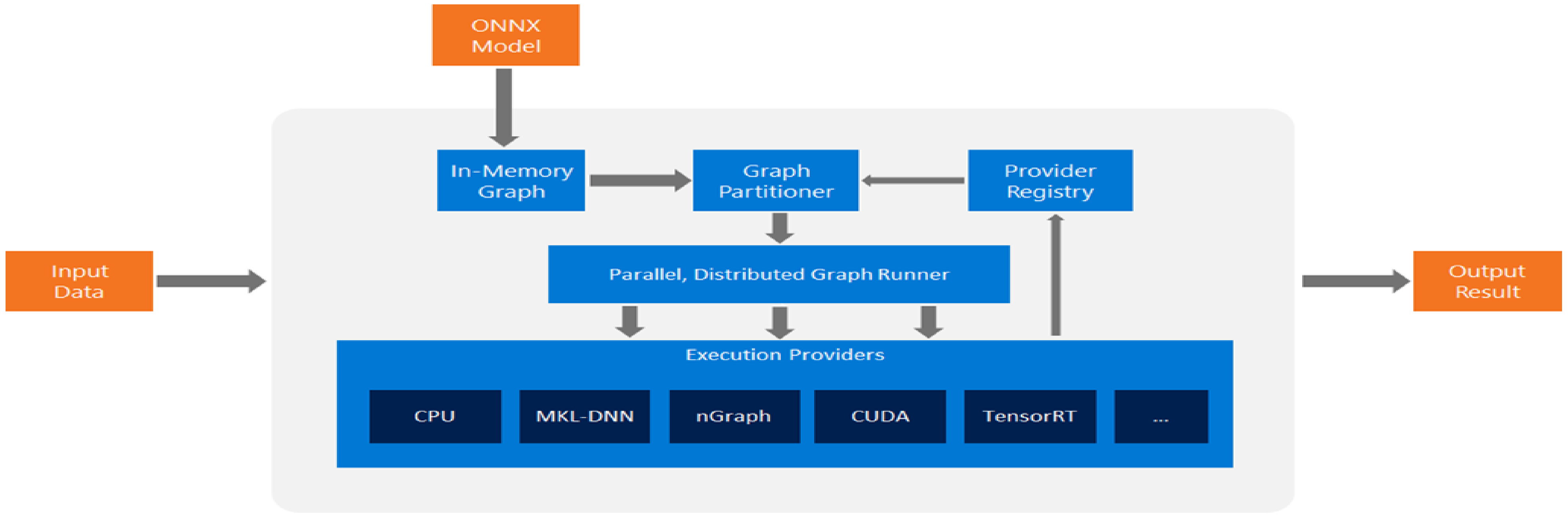
- Still under Microsoft…
- ONNX Runtime is a performance-focused inference engine for ONNX models.
- Perceived to be the de-facto runtime for running ONNX models, a solution for systems to integrate a single inference engine to support models trained from a variety of frameworks, while taking advantage of specific hardware accelerators where available.
- Designed with a focus on performance and scalability in order to support heavy workloads in high-scale production scenarios.
- Provides extensibility options for compatibility with emerging hardware developments.

ONNX Runtime: Key Features

- Cross Platform
  - The runtime provides a cross platform API compatible with Windows, Linux, and Mac and a variety of architectures. Both CPU and GPUs are supported, and language bindings are available for a variety of languages and architectures
- Run any ONNX model
  - ONNX Runtime provides comprehensive support of the ONNX spec and can be used to run all models based on ONNX v1.2.1 and higher. Both ONNX (DNN) and ONNX-ML (traditional ML) operator sets are supported.
- Backwards Compatible
  - Newer versions of ONNX Runtime support all models that worked with prior versions, so updates should not break integrations.

ONNX Runtime: System Architecture

Starting from an ONNX model, ONNXRuntime first converts the model graph into its in-memory graph representation. It then applies a number of graph transformations that a) perform a set of provider independent optimizations such cast transformations between float16 and float32, and b) partition the graph into a set of subgraphs based on the available execution providers. Each subgraph is assigned to an execution provider.

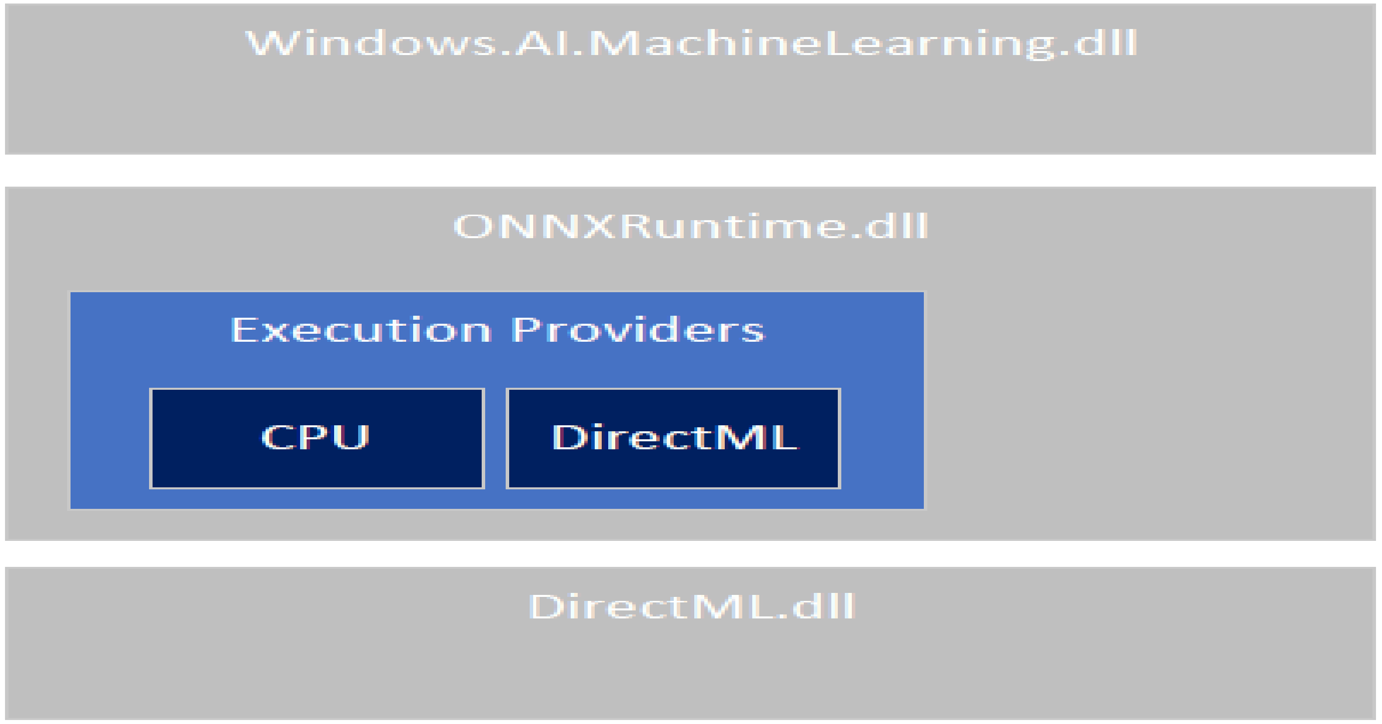


ONNX Runtime: Extensibility

- Implementations of the operators by execution providers are called kernels. Each execution provider supports a subset of the (ONNX) operators/kernels.
- The ONNX Runtime guarantees that all operators are supported by the default execution provider.
- ONNXRuntime utilizes a standard representation for the tensor runtime values. The execution providers can internally use a different representation and need to convert the values from/to the standard representation at the boundaries of their subgraph.
- Extensibility options
    - Add a custom operator/kernel
    - Add an execution provider
    - Add a new graph transform
    - Add a new rewrite rule

ONNX Runtime: Windows Integration

The ONNX runtime shipped with the Windows operating system. The runtime was embedded inside the Windows.AI.MachineLearning.dll and was exposed via WinRT API (WinML for short). It includes CPU support and a DirectML execution provider for GPU support.


Windows.AI.MachineLearning.dll
ONNXRuntime.dll
Execution Providers
CPU
DirectML
DirectML.dll