ONNX Introduction

Chin Huang, Cognitive OpenTech

Data and AI
Open Source Dojo

Outline

- Why and What is ONNX
- Overview and components
- ONNX ecosystem
- ONNX use cases

Challenges and Issues

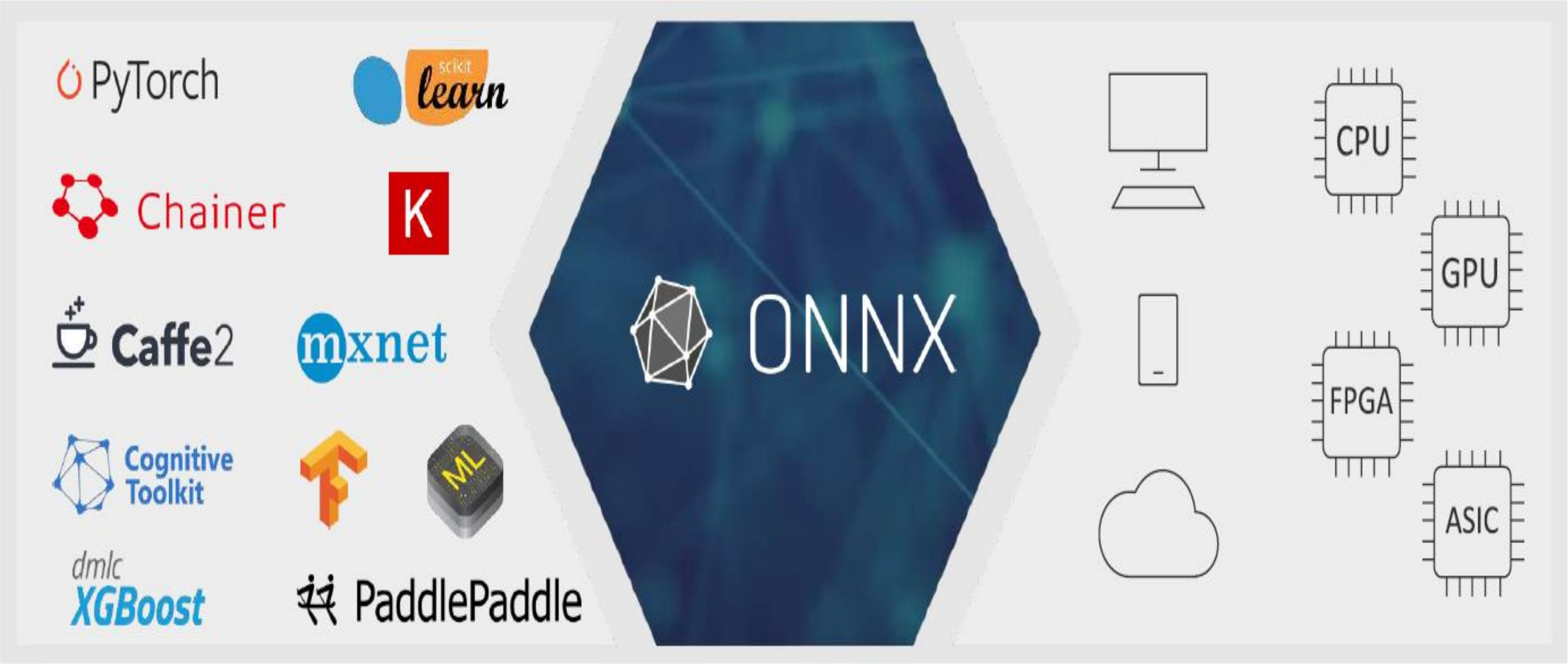
AI is booming!

- AI stacks evolved organically across organizations using a wide variety of frameworks
- Lots of target training and inferencing platforms
- Different hardware fabrics also evolved looking to meet training and serving needs

Challenges

- How to translate from one framework/platform combination to another?
- Difficult to share and re-use domain knowledge and applications
- Increasing fragmentation across companies and teams
- Long AI development cycle, idea -> research -> dev -> production -> maintain -> refine -> upgrade

Open and Interoperable AI



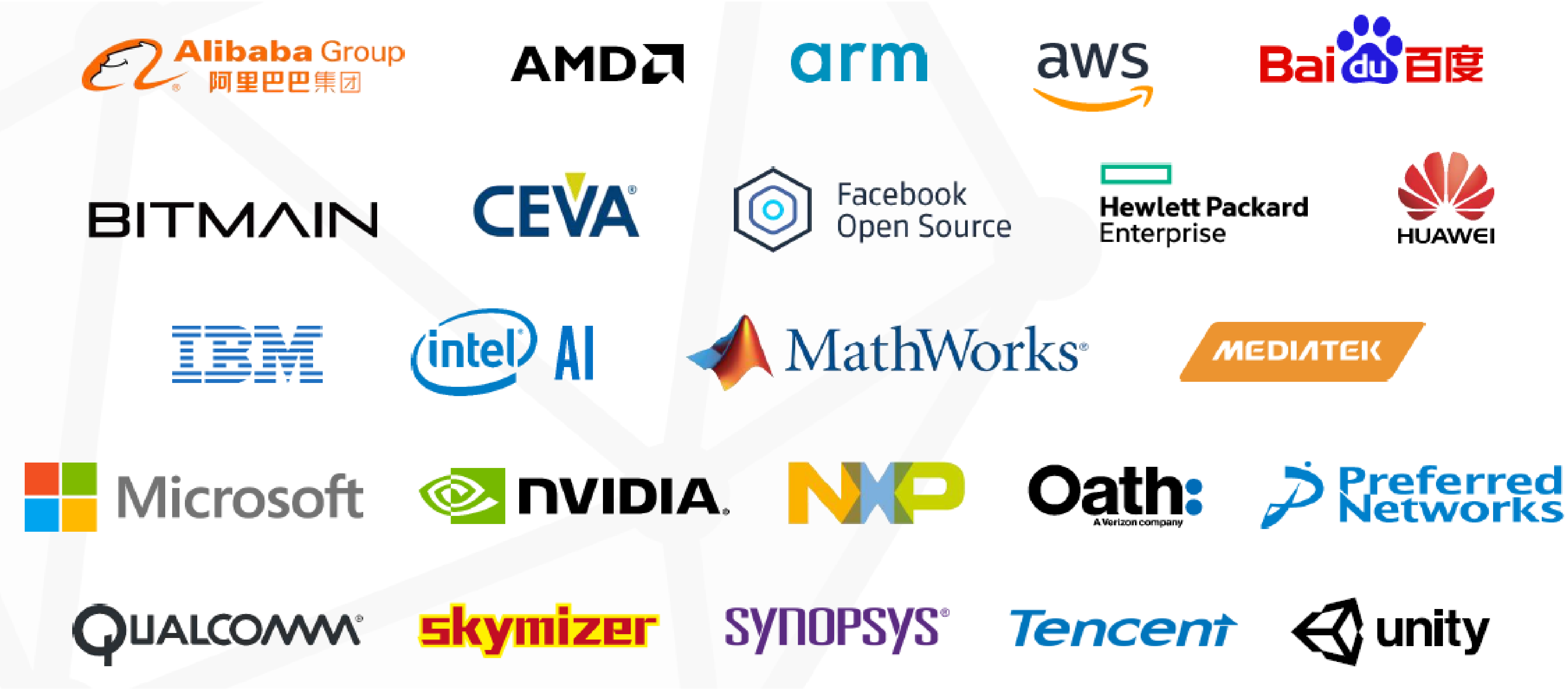
What is ONNX (Open Neural Network eXchange)?

- Open ecosystem for interchangeable AI models
- Designed to be an open specification and format, empowering developers to freely select the framework/tool that works best for their project, at any stage of development.
- Commonly describes the model graph, which serves as an Intermediate Representation (IR) that captures the specific intent of the developer's source code.
- Enables flexible graph optimization.
- Persisted as binary protobuf files. Each model file contains the network structure and parameters of the model.
- Advantages: open, framework neutral, cross-platform, complete ecosystem

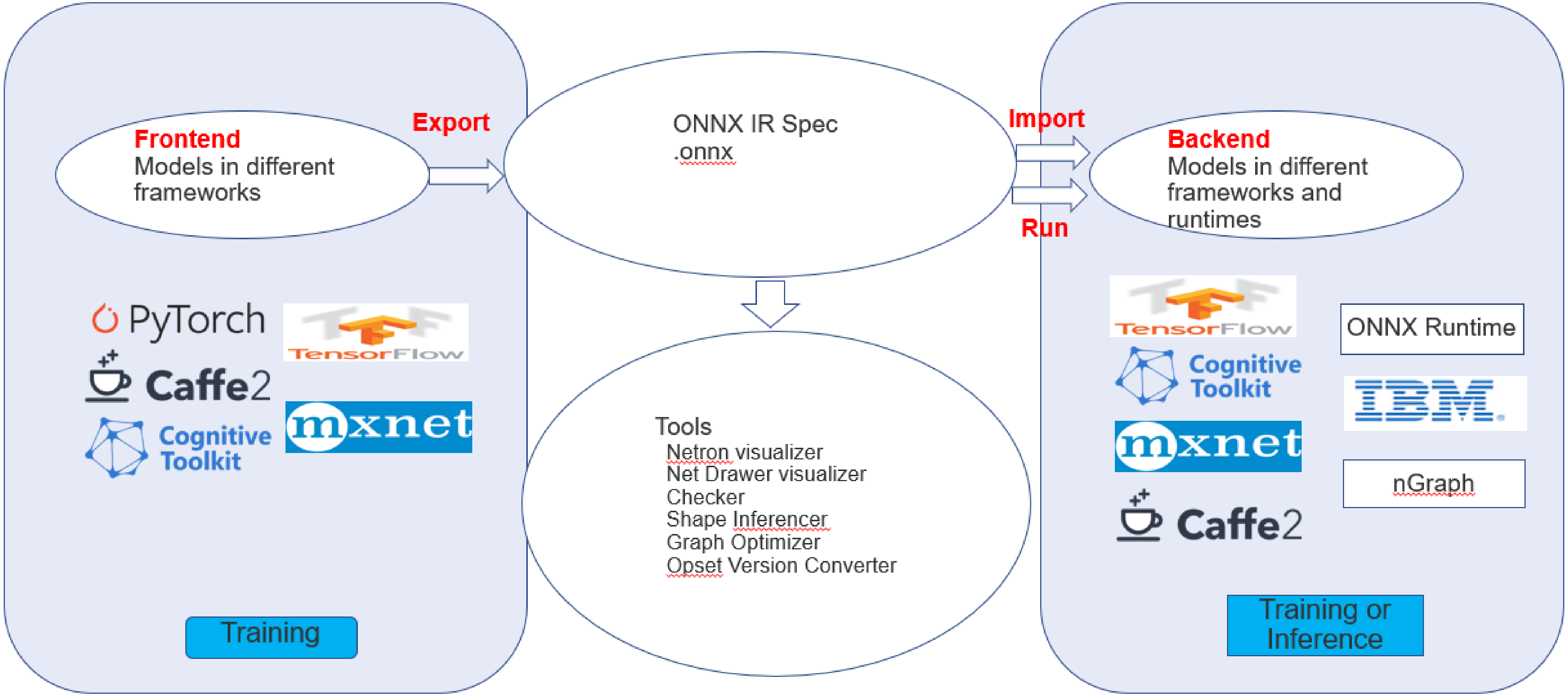
Background and factoids

- Started Sept 2017 by Microsoft and Facebook
- Initial goal is to make it easier for data analysts to exchange trained models between different machine learning frameworks.
- Model training is added in 2020 to describe trainable models
- ONNX github has 20 repos. onnx is the core. Others are tutorials, model zoo, importers and exporters for frameworks.
- Onnx/onnx currently has 15 releases, 153 contributors, 8015 stars.
- Core is in C++ with python API and tools.
- Supported frameworks: Caffe2, Chainer, Cognitive Toolkit (CNTK), Core ML, MXNet, PyTorch, PaddlePaddle, Tensorflow…

ONNX Partners



ONNX Overview



Frontend
Models in different frameworks

Export

ONNX IR Spec
.onnx

Import

Run

Backend
Models in different frameworks and runtimes

Tools
- Netron visualizer
- Net Drawer visualizer
- Checker
- Shape Inferencer
- Graph Optimizer
- Opset Version Converter

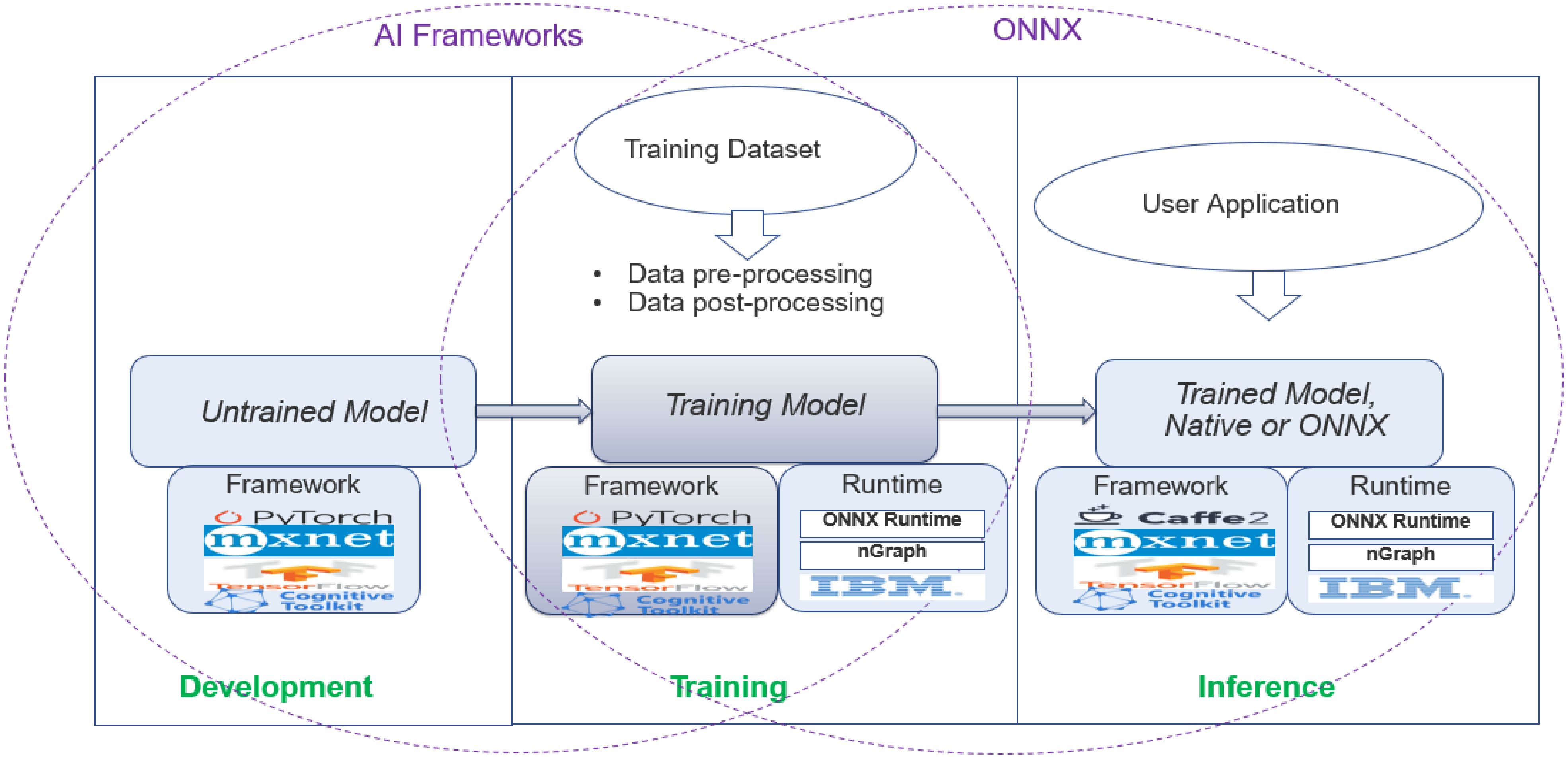
Training

ONNX Runtime

nGraph

Training or Inference

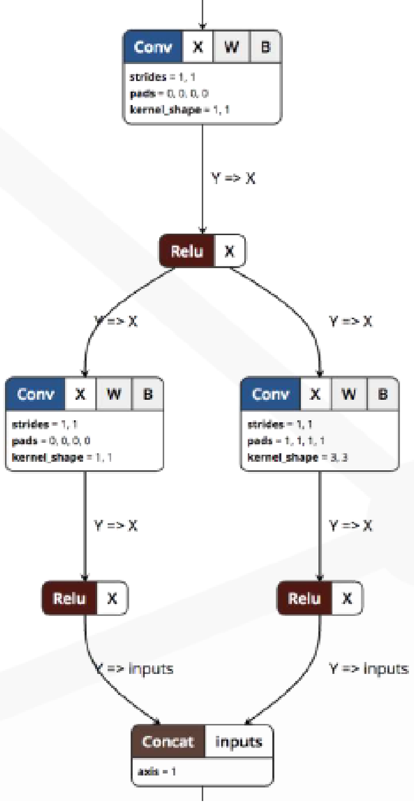
AI Model Development Cycle with ONNX



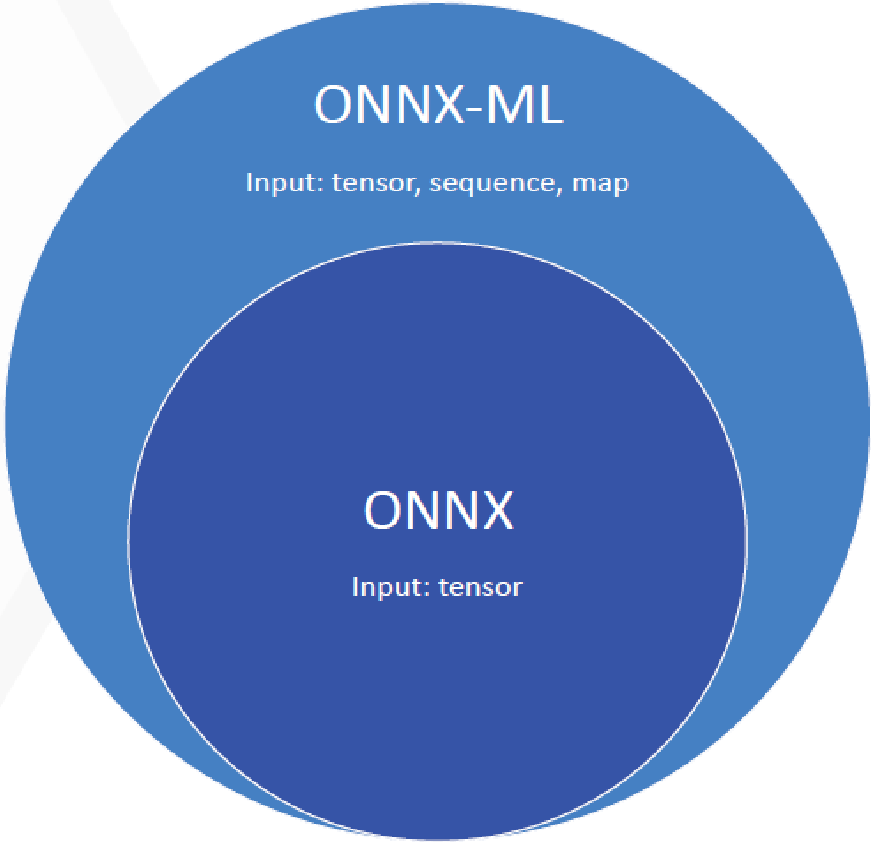
ONNX Core

ONNX is an open specification:

- Computation graph (IR)
- Data types
- Operators and functions



Two profiles:

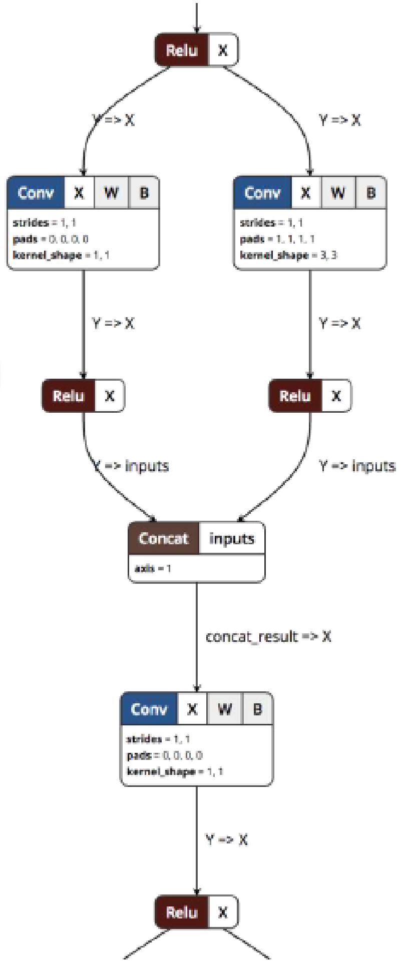


High Level IR

Model
- Version info
- Metadata
- Acyclic computation dataflow graph

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

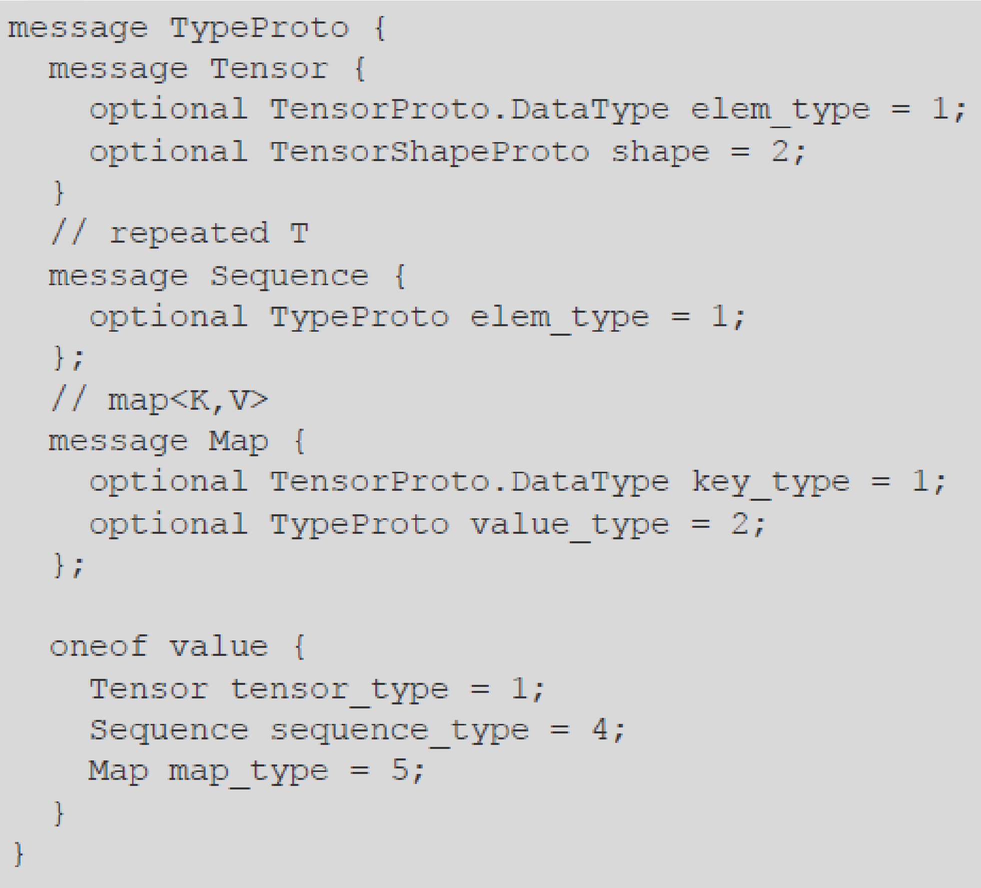
Computation Node
- Zero or more inputs of defined types
- One or more outputs of defined types
- Operator
- Operator parameters



Data Types

- Tensor type
  - Element types supported:
    - int8, int16, int32, int64
    - uint8, uint16, uint32, uint64
    - float16, float, double
    - bool
    - string
    - complex64, complex128
    - bfloat16

- Non-tensor types in ONNX-ML:
  - Sequence
  - Map



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

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

Operators

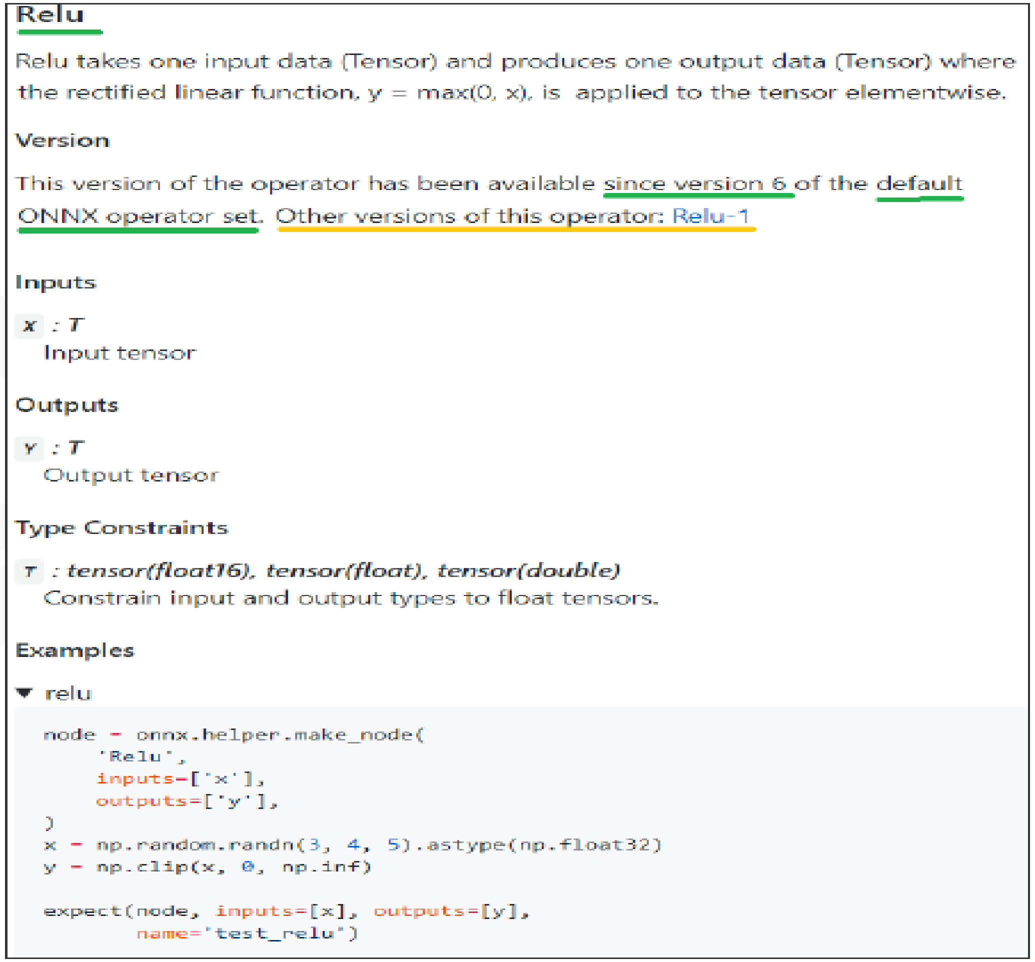
An operator is identified by <name, domain, version>

Core ops (ONNX, ONNX-ML, ONNX-Training domains)

- Should be supported by all ONNX-compatible products
- Generally cannot be meaningfully further decomposed
- Version incremented for breaking changes
- ONNX-Training ops for model training only

Custom ops

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



Relu

Relu takes one input data (Tensor) and produces one output data (Tensor) where the rectified linear function, y = max(0, x), is applied to the tensor elementwise.

Version

This version of the operator has been available since version 6 of the default ONNX operator set. Other versions of this operator: Relu-1

Inputs

x : *T*
Input tensor

Outputs

Y : *T*
Output tensor

Type Constraints

T : *tensor(float16), tensor(float), tensor(double)*
Constrain input and output types to float tensors.

Examples

▼ relu

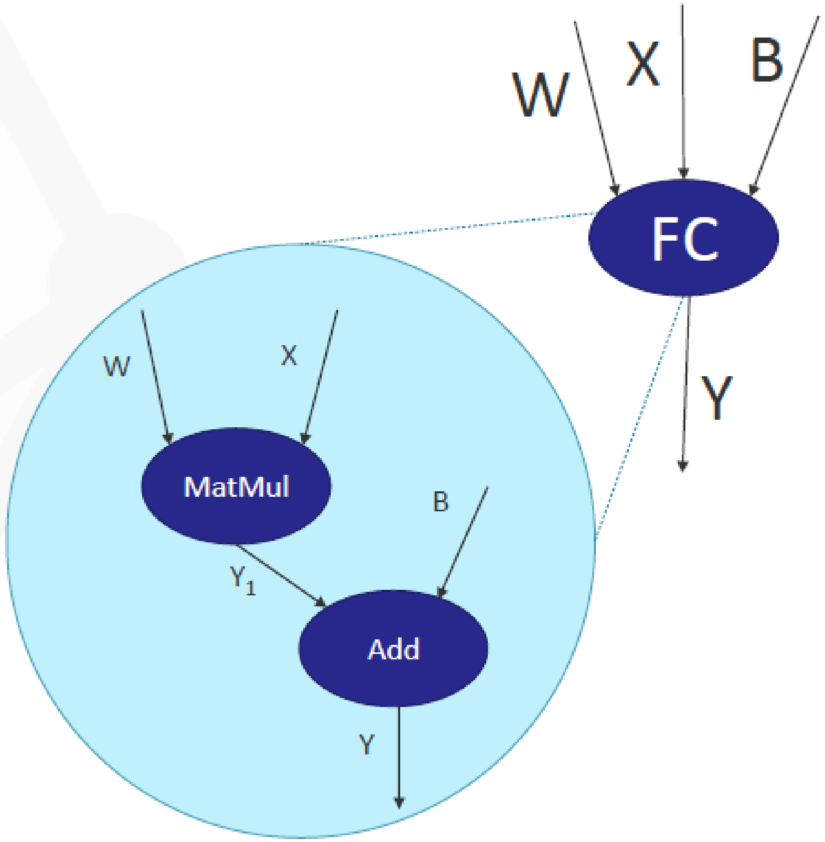
```
node = onnx.helper.make_node(
    'Relu',
    inputs=['x'],
    outputs=['y'],
)
x = np.random.randn(3, 4, 5).astype(np.float32)
y = np.clip(x, 0, np.inf)

expect(node, inputs=[x], outputs=[y],
       name='test_relu')
```

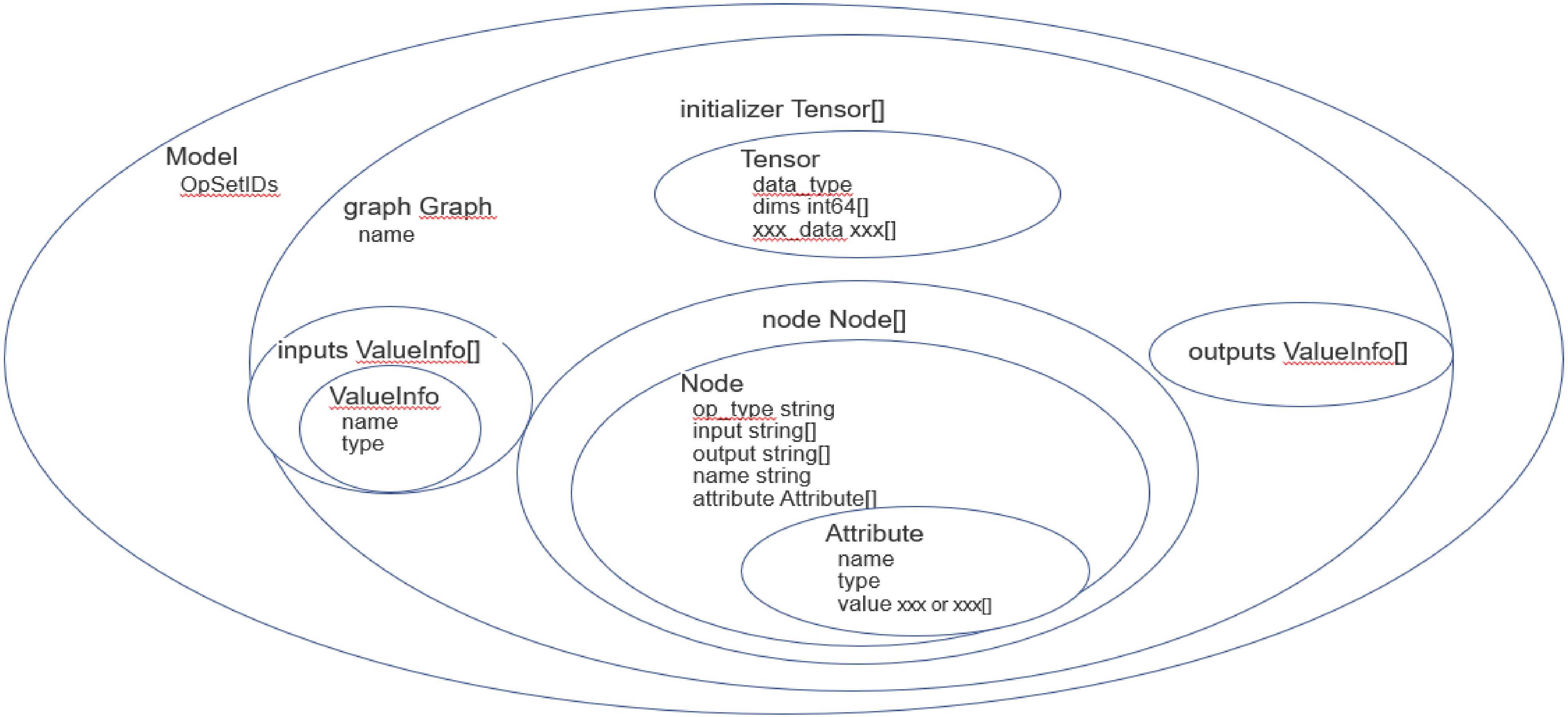
Functions

- Compound ops built with existing primitive ops
- Enable potential optimizations by runtimes, e.g. node fusion

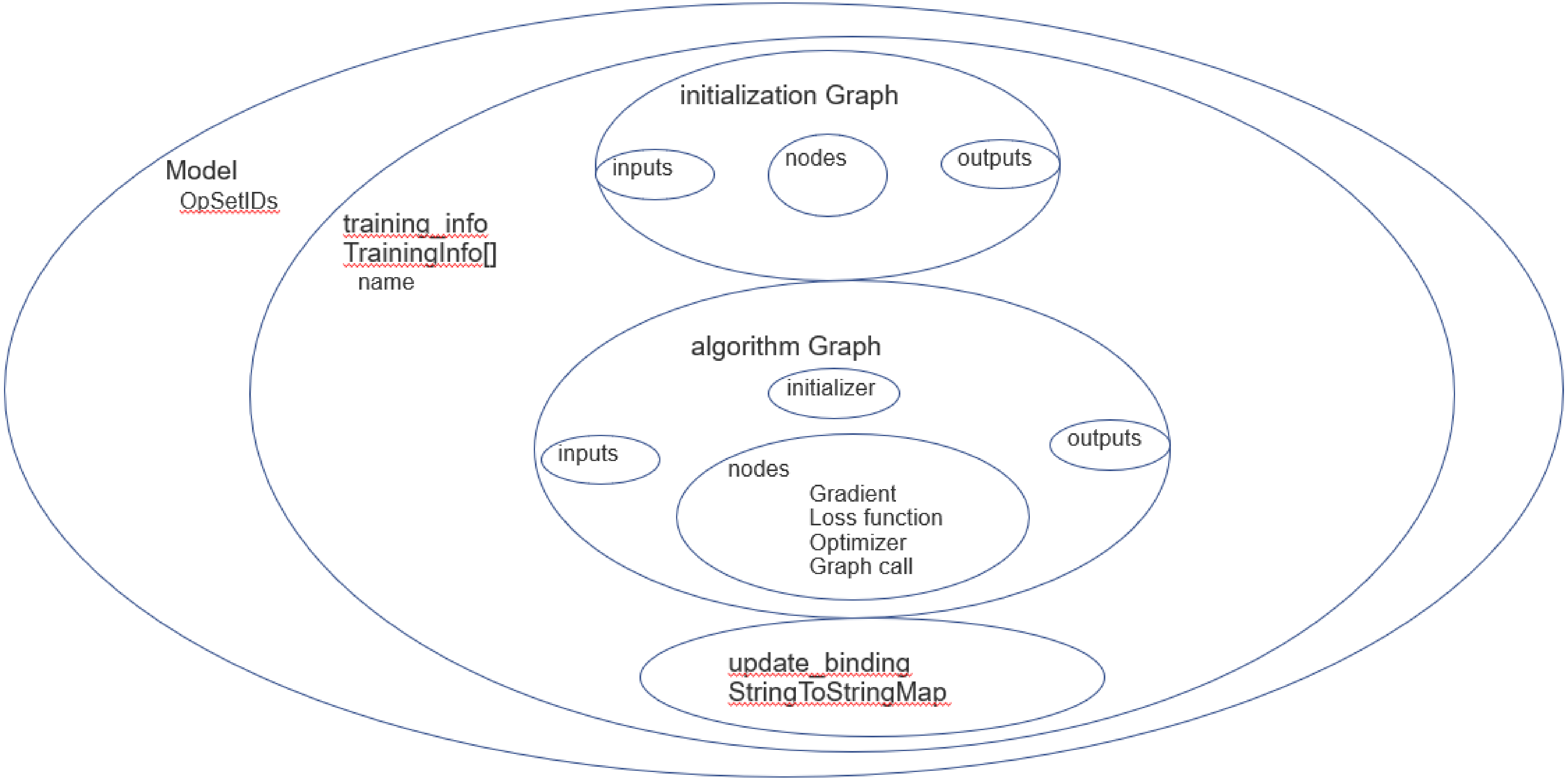
- Two types of functions:
  - Standardized functions - defined by ONNX spec
  - In-model customized functions (not in spec yet) Defined within the model itself; runtimes/frameworks/tools that don't know the function can fallback to use the primitive ops



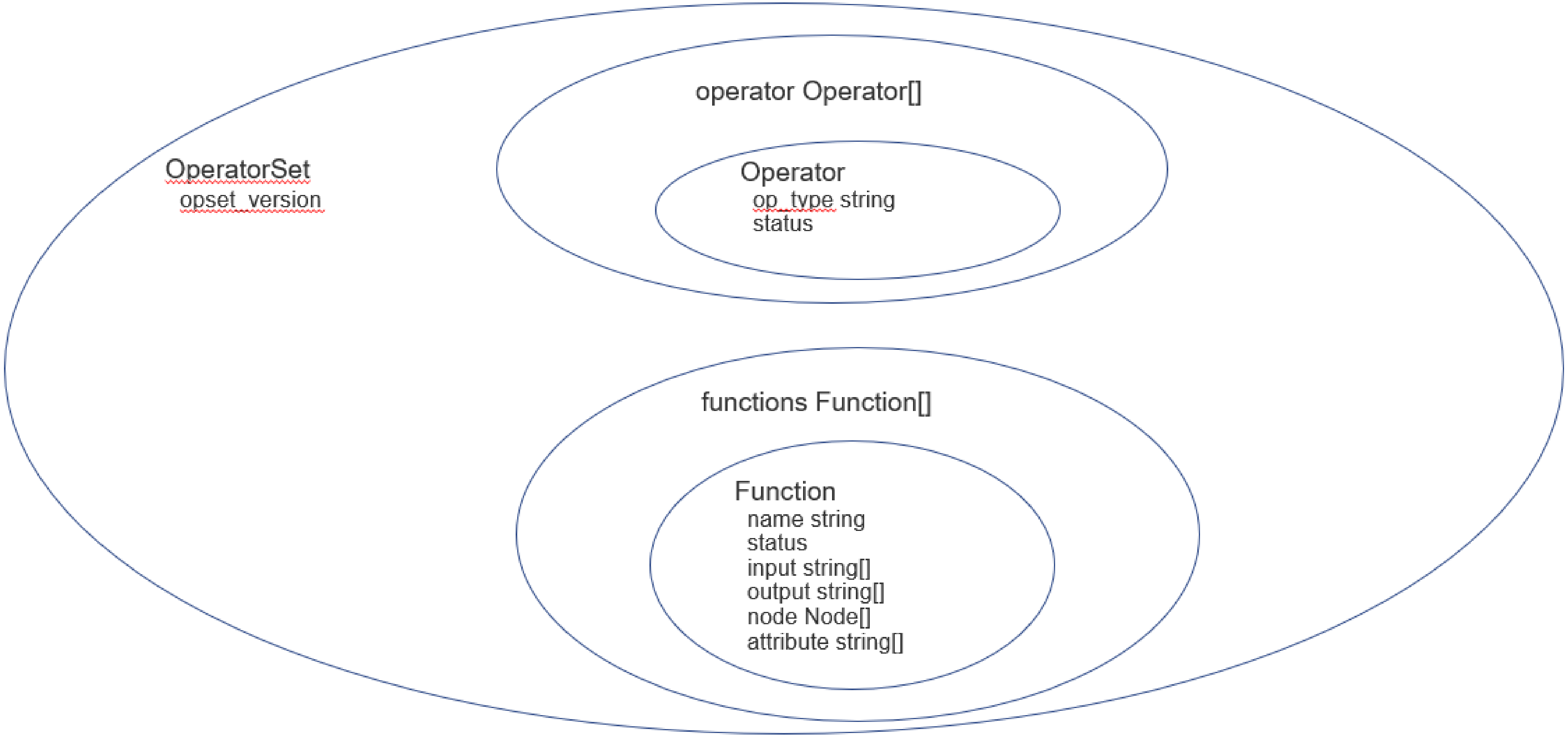
ONNX IR Spec – Computation Graph



ONNX IR Spec – Training Graph



ONNX IR Spec – Operator Set



OperatorSet
opset_version

operator Operator[]

Operator
op_type string
status

functions Function[]

Function
name string
status
input string[]
output string[]
node Node[]
attribute string[]

ONNX Backend

- An ONNX backend is a library that can run ONNX models.
- Each AI framework can create a converter that converts ONNX models from and to the corresponding framework specific representation and then delegate the execution to the framework.
- ONNX has defined a unified (Python) backend interface at https://github.com/onnx/onnx/blob/master/onnx/backend/base.py
- ONNX provides a standard backend test suite to assist backend implementation verification. Two types of test are 1. nodes tests: verify whether a backend is performing the correct computation 2. model tests, verify the backend at the model level. It's strongly encouraged that each ONNX backend runs this test.

ONNX Tools

- Visualizer
  - Net Drawer
  - Netron (https://github.com/lutzroeder/netron)
- Checker: validate ONNX models
- Optimizer: perform optimizations on ONNX models
- Shape inferencer: query the shape of tensor
- Version converter: convert ONNX models between different opset versions

ONNX Optimizer

- ONNX provides a C++ library for performing arbitrary optimizations on ONNX models, as well as a growing list of prepackaged optimization passes.
- The primary motivation is to share work between the many ONNX backend implementations.
- The aim is to provide all such passes along with ONNX so that they can be re-used with a single function call.
- The API Optimize() accepts an input ModelProto and a list of optimization passes to apply, and returns a new ModelProto which is the result of applying each of the named passes in sequence to the model.

ONNX Interface for Framework Integration (ONNXIFI)

- A cross-platform API for loading and executing ONNX graphs on optimized backends
- Features:
  - Standardized interface for neural network inference on special-purpose accelerators, CPUs, GPUs, DSPs, and FPGAs
  - Dynamic discovery of available backends for model execution
  - Dynamic discovery of supported ONNX Operators on each backend
  - Extensible interface for vendor-specific operators and vender-provided ONNXIFI implementation

Python API

- ONNX model lifecycle support
  - Loading, saving, creating models
  - Checking/validating
  - Optimizing
  - Running shape inference
  - Converting versions
  - Polishing (runs model checker, optimizer, shape inference engine on the model, and also strips the doc_string)

ONNX Use Cases

- Microsoft

PLATFORMS

AzureML

WinML

ML.NET

ML.Net

PRODUCTS

Bing

Microsoft Cognitive Services

Office 365

ads

Power BI

skype

Up to 14.6x

Performance gains seen by Microsoft services

100s of Millions

# of devices where ONNX Runtime is running

Billions

# of requests handled by ONNX Runtime across Microsoft services

ONNX Use Cases

- Huawei
  - MindSpore: a full-stack AI framework open sourced in 1Q 2020
  - MindSpore IR -> ONNX model -> ONNX Runtime
  - ONNX model -> MindSpore IR, optimized for Ascent chips
- MathWorks (MATLAB)
  - Two-way model exchange with ONNX <--> MATLAB
  - Automatic code gen, visualization, re-training in MATLAB
  - Goals: Import 90% of ONNX model zoo models to MATLB, export 90% of MATLAB models to ONNX!

ONNX Use Cases

- Intel OpenVINO Toolkit
  - Visual Inference & Neural Network Optimization, computer vision toolkit for edge computing
  - Use case 1: direct convert from ONNX models
  - Use case 2: as an ONNX Runtime EP (execution provider)
- Tencent (Chinese social platforms such as QQ and WeChat/Weixin)
  - https://github.com/Tencent/ncnn
  - NCNN deploys onnx for mobile, supporting Arm and GPUs
  - Sample case: pytorch -> onnx -> ncnn

ONNX Use Cases

- Alibaba
  - SinianAI: a full-stack AI framework, acceleration for Cloud, Edge, and IoT
  - Use cases: smoking detection, self checkout, voice activation
- UC Santa Cruz
  - Deep neural network using ONNX for efficient genome analysis
  - Global effort with multiple cross-country research consortiums

ONNX at IBM

- Open Source Contributions
  - ONNX
    - Converters SIG lead
    - Training working group lead
    - Power CI build and test
    - Release manager for 1.7
  - ONNX-TensorFlow converter
  - ONNX-MLIR lead
- Open Governance: Took ONNX to LF AI as a graduated project!

ONNX at IBM

- IBM Products and Services
  - IBM Watson Machine Learning for Z
  - IBM Watson Machine Learning Accelerator