Leverage Power of Machine Learning with ONNX

Ron Dagdag

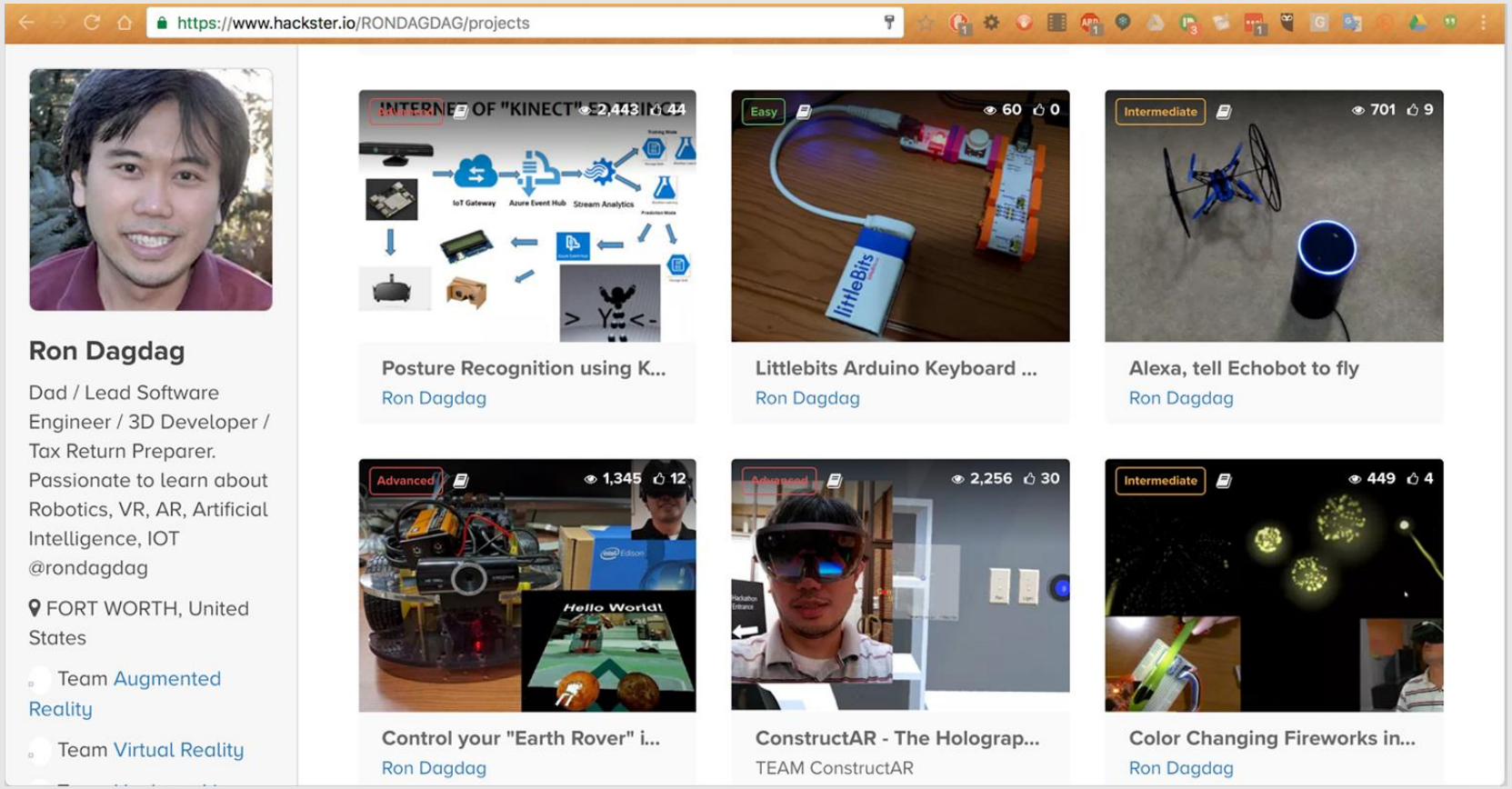
@rondagdag

http://bit.ly/ml-onnx

Hackster Portfolio

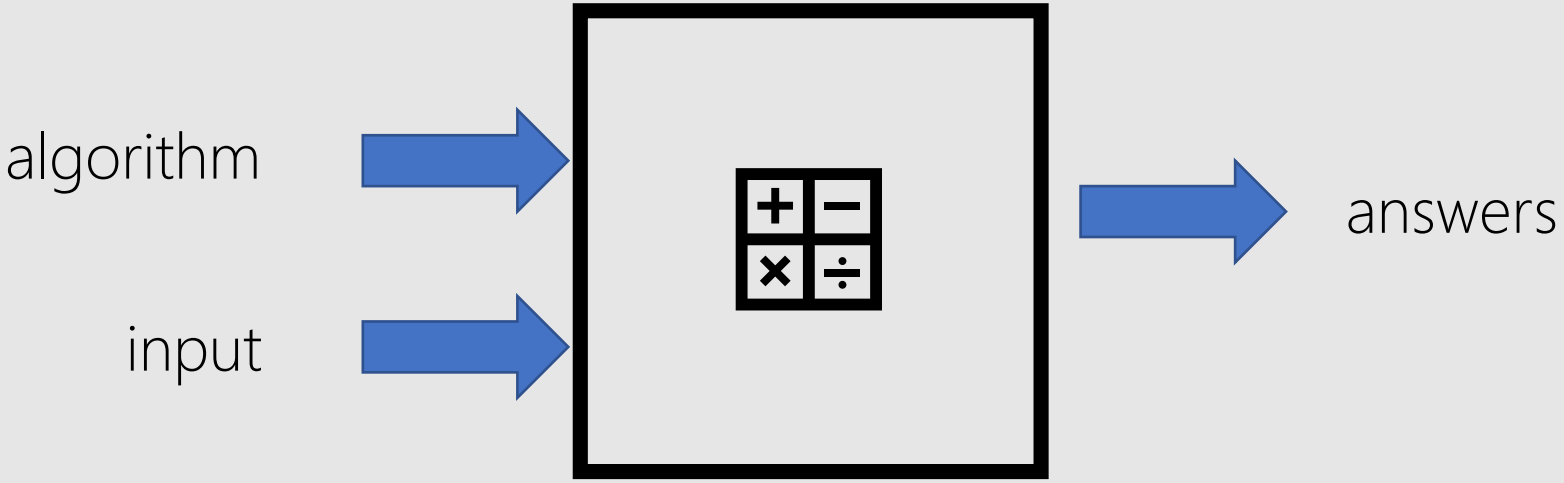
www.dagdag.net

@rondagdag

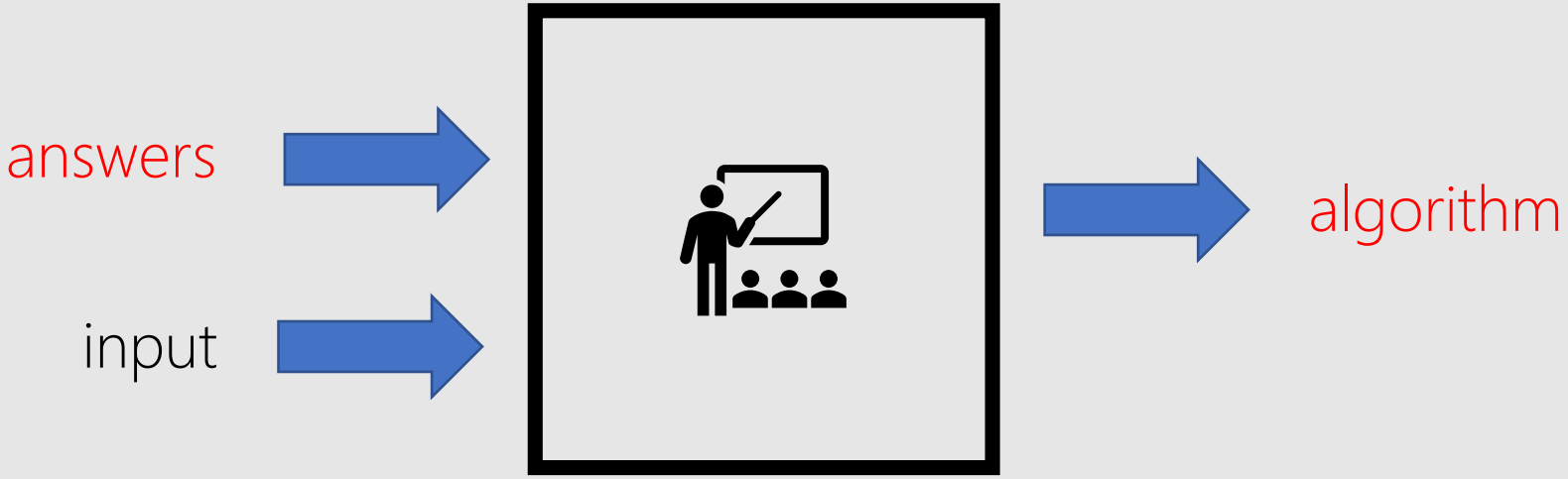


ONNX, Not ONIX

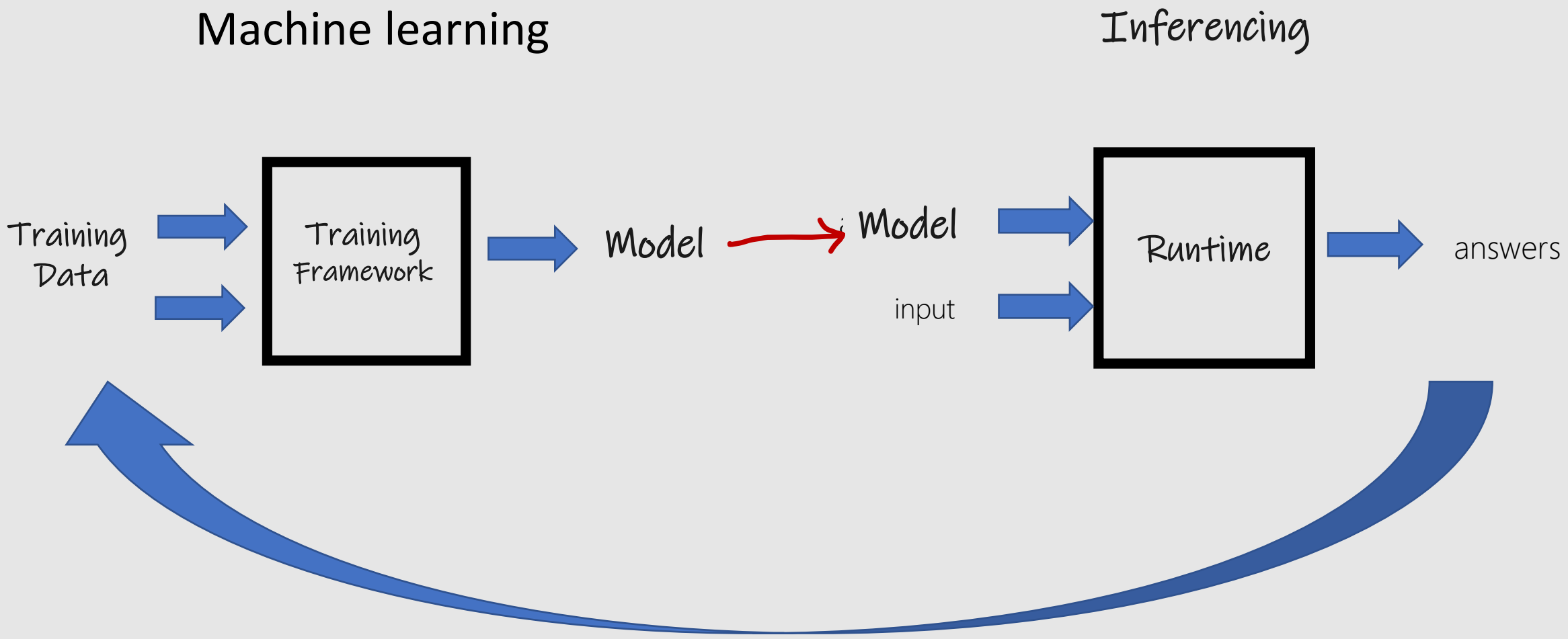
programming



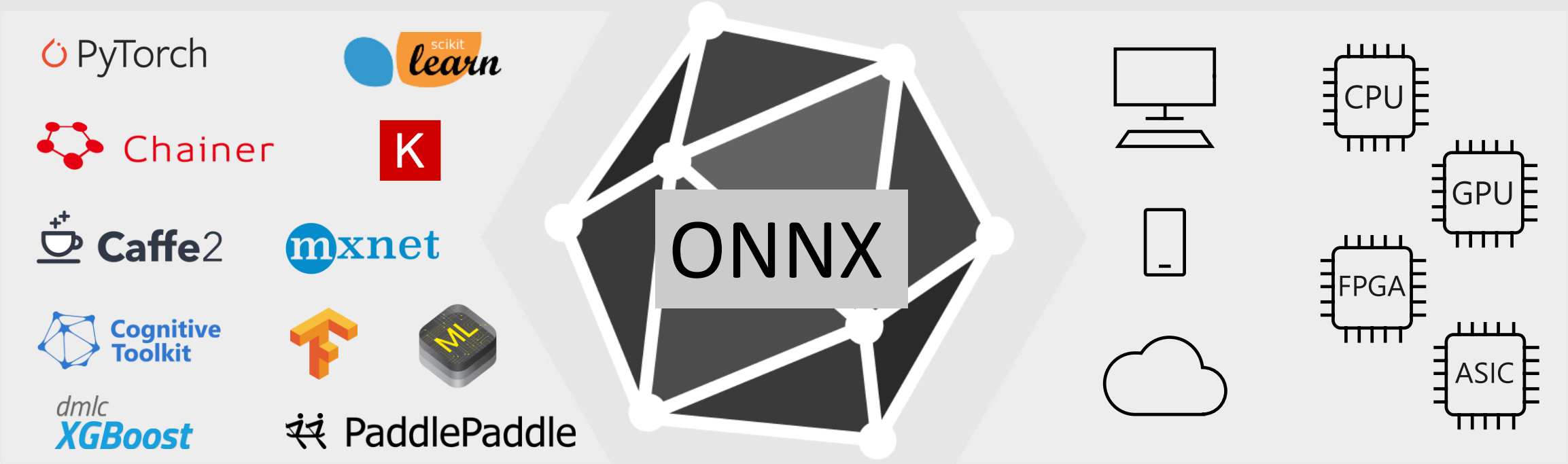
machine learning



ML Primer


Machine learning
Inferencing
Training Data
Training Framework
Model
Model
input
Runtime
answers

Open and Interoperable AI



ONNX

Open Neural Network Exchange

Open format for ML models

github.com/onnx

onnx.ai/

ONNX Partners

ABBYY
Alibaba Group 阿里巴巴集团
AMD
arm
aws
Baidu 百度
BECKHOFF
BITMAIN
cadence
CEVA
Facebook Open Source
GRAPHCORE
habana
Hewlett Packard Enterprise
HUAWEI
IBM
Idein Inc
intel AI
MathWorks
MAXAR
MEDIATEK
mi
Microsoft
NVIDIA
NXP
OctoML
OPEN AI LAB 开放智能
Preferred Networks
SIEMENS
SONY
Qualcomm
sas
商汤 sensetime
skymizer
SYNOPSYS
Tencent
unity
verizon media
WOLFRAM
Yandex

9.2k Github Stars

~1700 Pull Requests

175 Contributors

1.6k Github Forks

36 Model Zoo

Agenda

- ✓ What is ONNX
- ❑ How to create ONNX models
- ❑ How to deploy ONNX models

Create

Frameworks

Caffe2 · Chainer · Cognitive Toolkit · mxnet · PyTorch · PaddlePaddle · ML.NET · MathWorks · dmlc XGBoost · ML · scikit learn · TF · K

Services

Azure Custom Vision Service

Native support

Converters

Native support

ONNX Model

Native support

Converters

Deploy

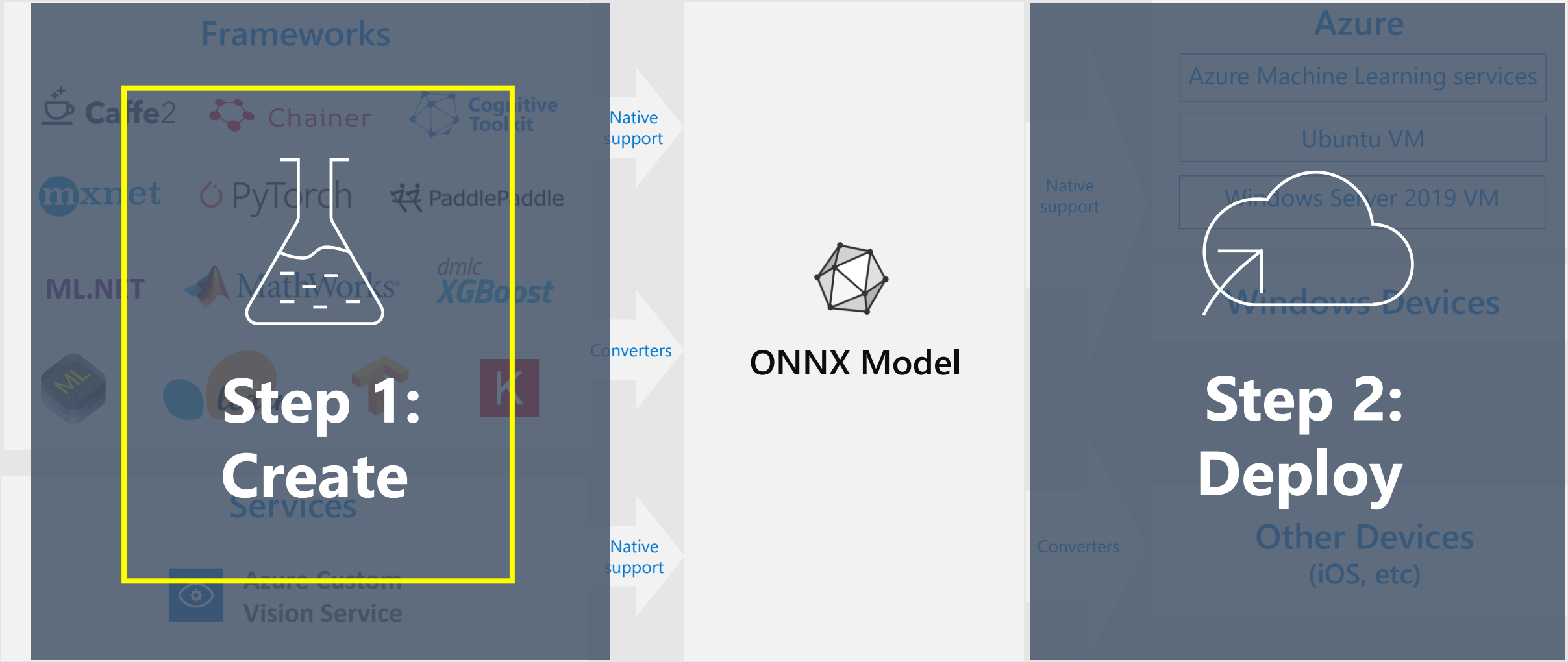
Cloud Services

- Azure Machine Learning services
- Ubuntu VM
- Windows Server 2019 VM

Windows Devices

IoT Edge Devices

Other Devices
(iOS, Android, etc)


Frameworks
Caffe2
Chainer
Cognitive Toolkit
mxnet
PyTorch
PaddlePaddle
ML.NET
MathWorks
dmlc XGBoost
Services
Azure Custom Vision Service
Step 1: Create
Native support
Converters
Native support
ONNX Model
Azure
Azure Machine Learning services
Ubuntu VM
Windows Server 2019 VM
Native support
Windows Devices
Converters
Other Devices (iOS, etc)
Step 2: Deploy

Secret Recipe

4 ways to get an ONNX model

ONNX Model Zoo

Azure Custom Vision Service

Convert existing models

Train models in Azure Machine Learning

Automated Machine Learning

ONNX Model Zoo: github.com/onnx/models

Image Classification

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

Model Class	Reference	Description
MobileNet	Sandler et al.	Efficient CNN model for mobile and embedded vision applications. Top-5 error from paper - ~10%
ResNet	He et al., He et al.	Very deep CNN model (up to 152 layers), won the ImageNet Challenge in 2015. Top-5
SqueezeNet	Iandola et al.	A ligh fewer Top-5
VGG	Simonyan et al.	Deep Challe Top-5

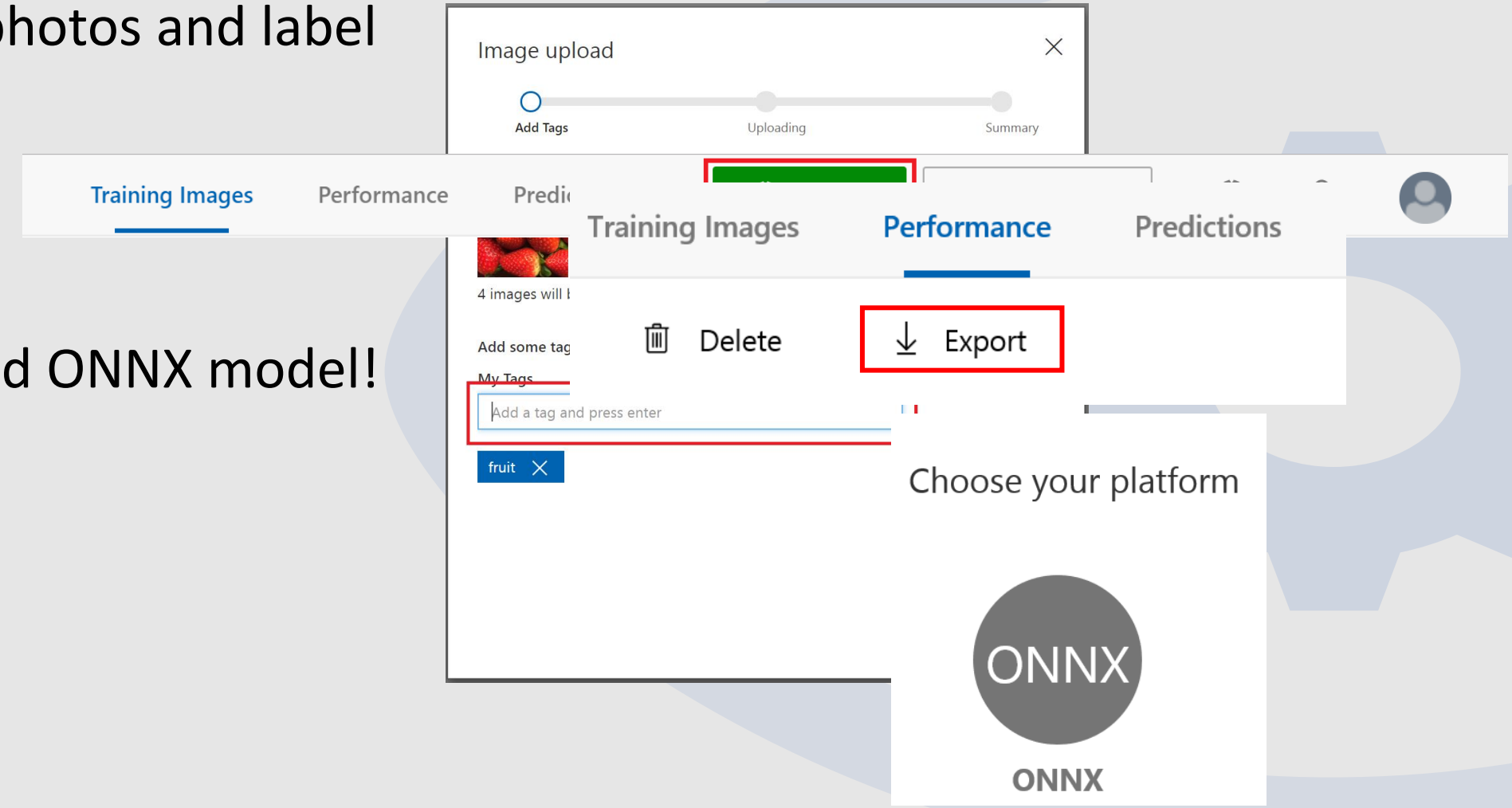
Model	Download	Checksum	Download (with sample test data)	ONNX version	Opset version	Top-1 accuracy (%)	Top-5 accuracy (%)
ResNet-18	44.6 MB	MD5	42.9 MB	1.2.1	7	69.70	89.49
ResNet-34	83.2 MB	MD5	78.6 MB	1.2.1	7	73.36	91.43
ResNet-50	97.7 MB	MD5	92.0 MB	1.2.1	7	75.81	92.82
ResNet-101	170.4 MB	MD5	159.4 MB	1.2.1	7	77.42	93.61
ResNet-152	230.3 MB	MD5	216.0 MB	1.2.1	7	78.20	94.21

Custom Vision Service: customvision.ai

1. Upload photos and label

2. Train

3. Download ONNX model!



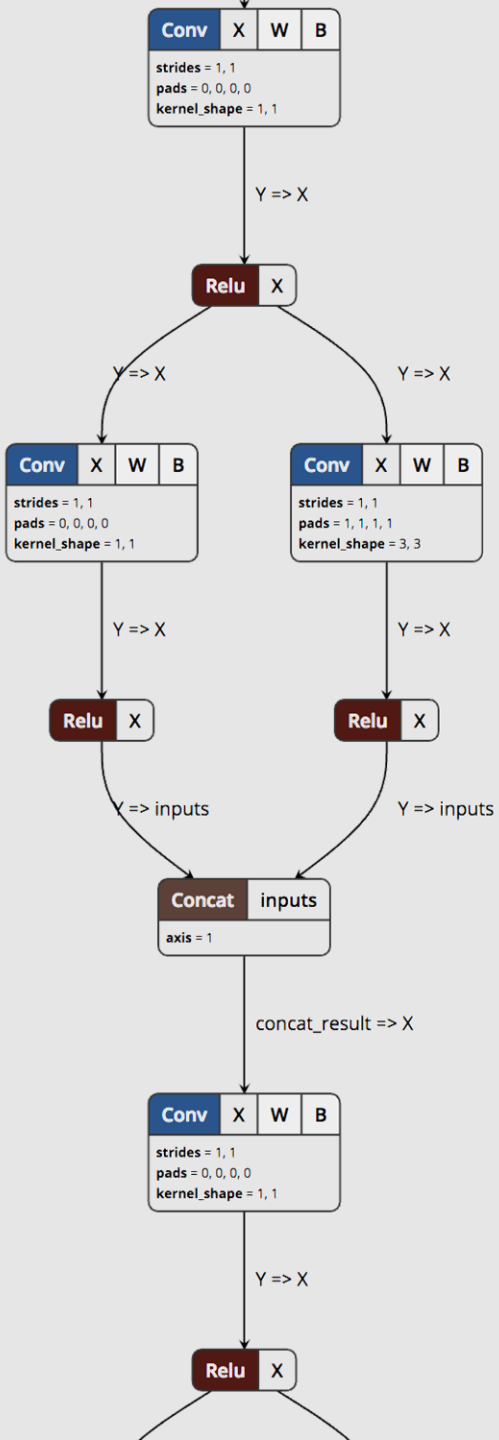
Convert models

Convert models

1. Load existing model
2. (Convert to ONNX)
3. Save ONNX model

ONNX Models

Graph of operations



Netron

https://lutzroeder.github.io/netron/

Convert models: Pytorch

```
import torch
import torchvision

dummy_input = torch.randn(10, 3, 224, 224, device='cuda')
model = torchvision.models.alexnet(pretrained=True).cuda()

input_names = [ "actual_input_1" ] + [ "learned_%d" % i for i in range(16) ]
output_names = [ "output1" ]

torch.onnx.export(model, dummy_input, "alexnet.onnx", verbose=True,
input_names=input_names, output_names=output_names)
```

Convert models: Keras

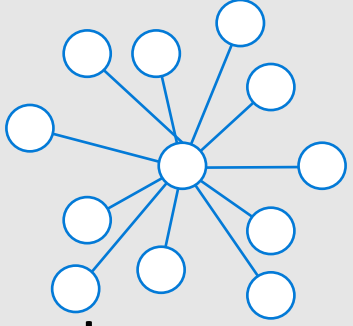
```
import numpy as np
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input
import keras2onnx
import onnxruntime

# load keras model
from keras.applications.resnet50 import ResNet50
model = ResNet50(include_top=True, weights='imagenet')

# convert to onnx model
onnx_model = keras2onnx.convert_keras(model, model.name)
```

Convert models: TensorFlow

python -m tf2onnx.convert --saved-model **tensorflow-model-path** --output **model.onnx**

https://github.com/onnx/tensorflow-onnx

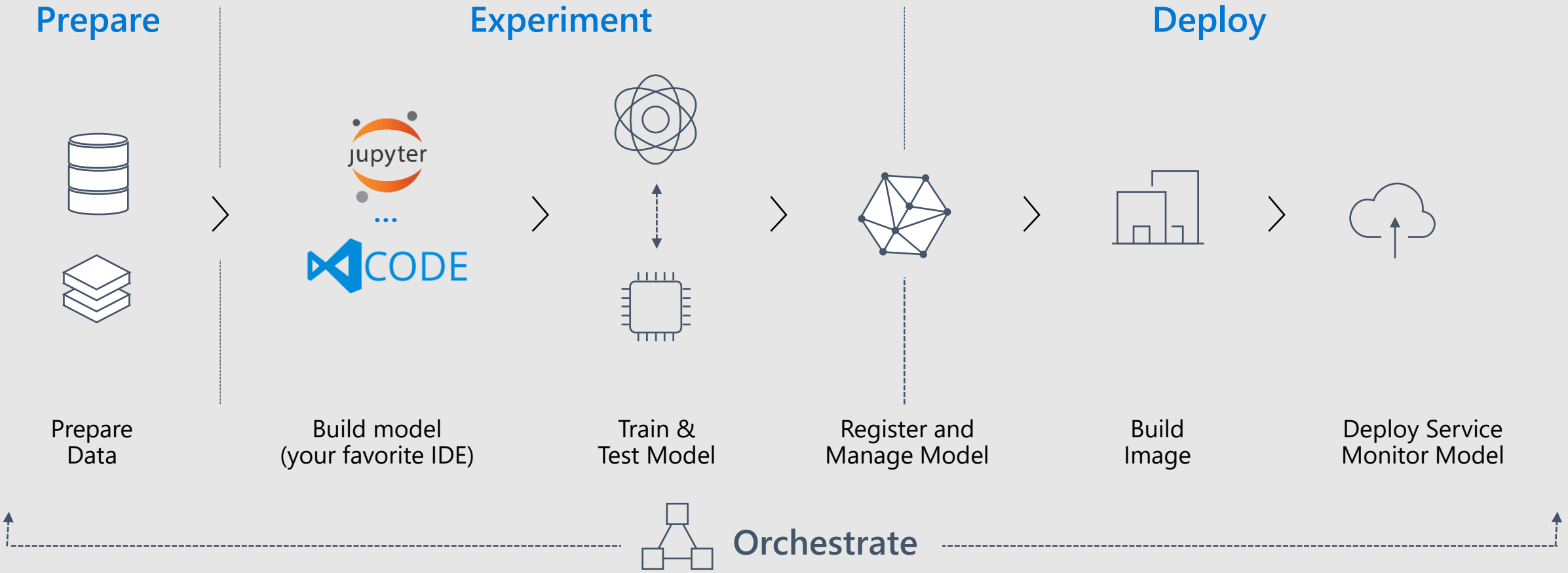
Train models in Azure Machine Learning



- Experiment locally then quickly scale with GPU clusters in the cloud
- Use automated machine learning and hyper-parameter tuning.
- Keeping Track of experiments, manage models, and easily deploy with integrated CI/CD tooling

Machine Learning

Typical E2E Process



high dimensional matrices

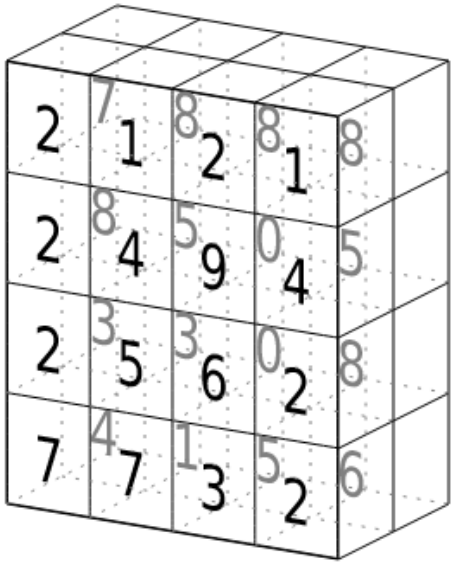
tensor

't'
'e'
'n'
's'
'o'
'r'

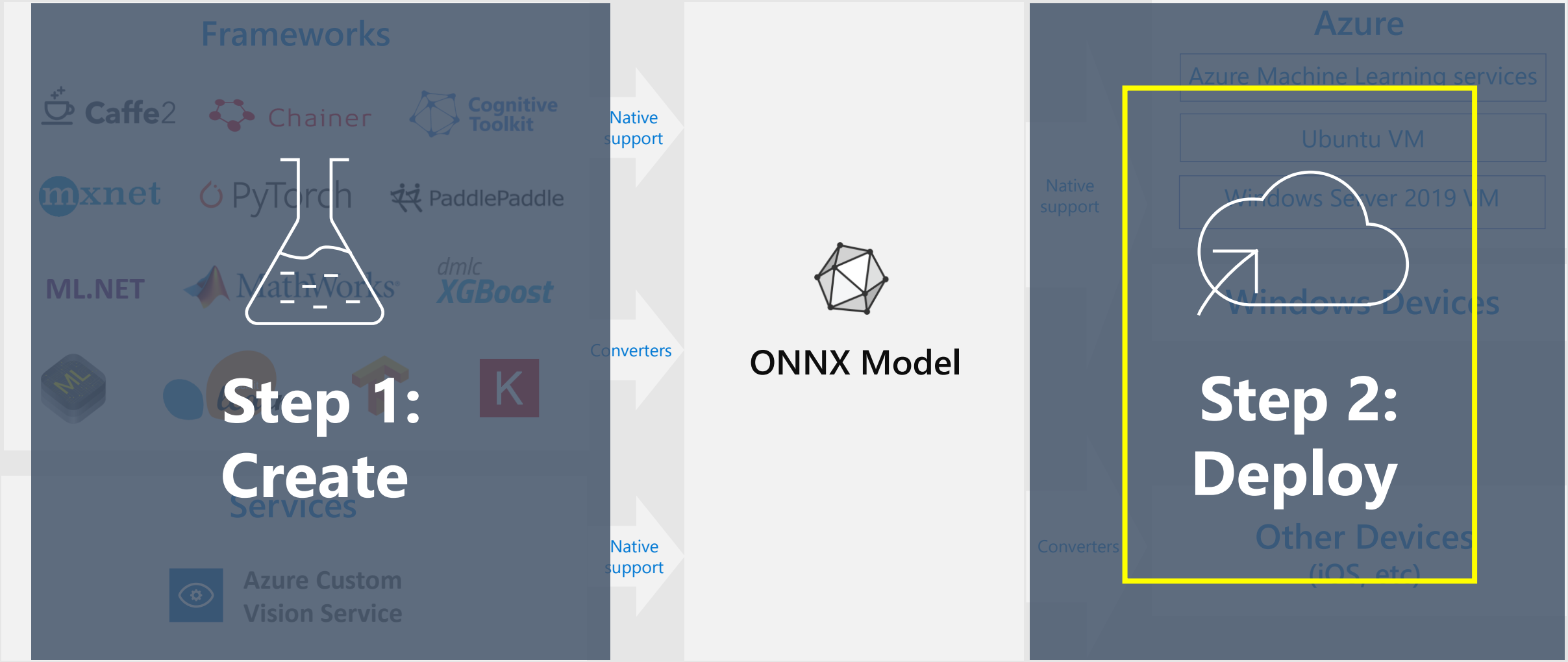
tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]


Frameworks
Caffe2
Chainer
Cognitive Toolkit
mxnet
PyTorch
PaddlePaddle
ML.NET
MathWorks
dmlc XGBoost
Step 1: Create
Services
Azure Custom Vision Service
Native support
Converters
Native support
ONNX Model
Azure
Azure Machine Learning services
Ubuntu VM
Windows Server 2019 VM
Native support
Windows Devices
Step 2: Deploy
Converters
Other Devices (iOS, etc)

Baker vs Starting a Bakery

Create

Frameworks

Caffe2 · Chainer · Cognitive Toolkit · mxnet · PyTorch · PaddlePaddle · ML.NET · MathWorks · dmlc XGBoost · ML · scikit learn · K

Services

Azure Custom Vision Service

Native support

Converters

Native support

ONNX Model

Native support

Converters

Deploy

Azure

- Azure Machine Learning services
- Ubuntu VM
- Windows Server 2019 VM

Windows Devices

IoT Edge Devices

Other Devices
(iOS, etc)

Cloud or Edge

Cloud or Edge

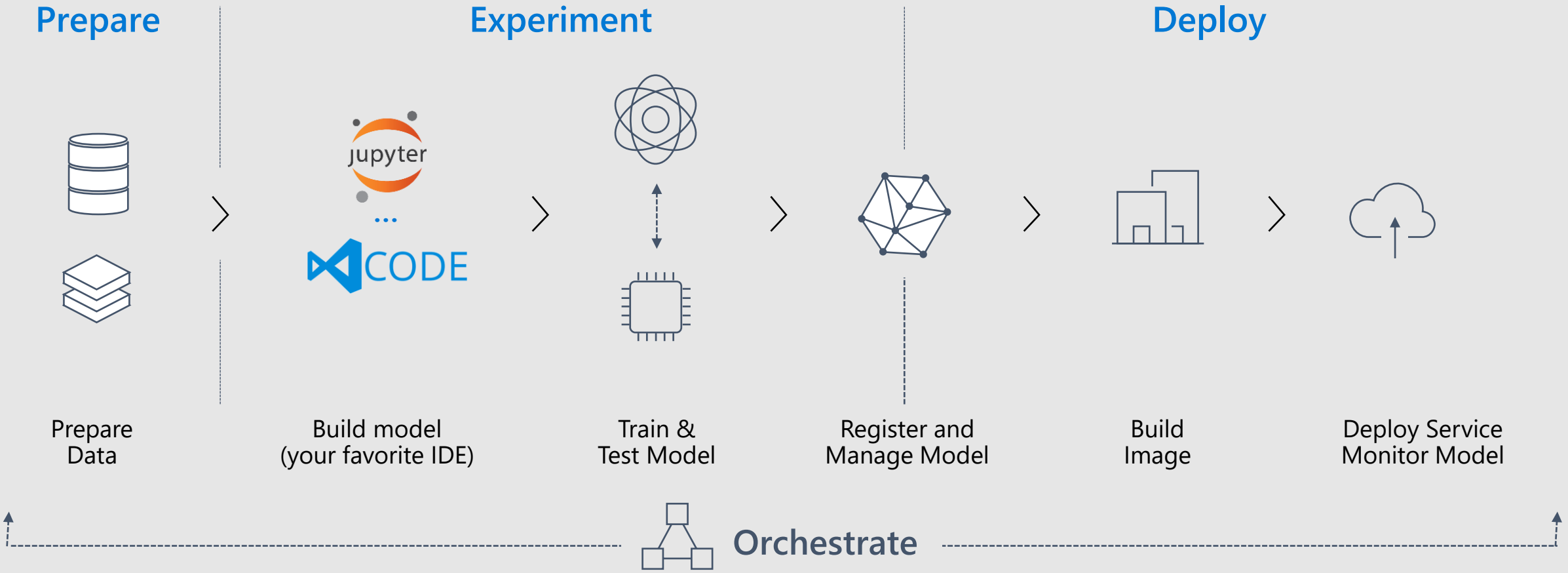
Deploy with Azure Machine Learning

- Model management services
- Deploy as web service to ACI or AKS
- Capture model telemetry

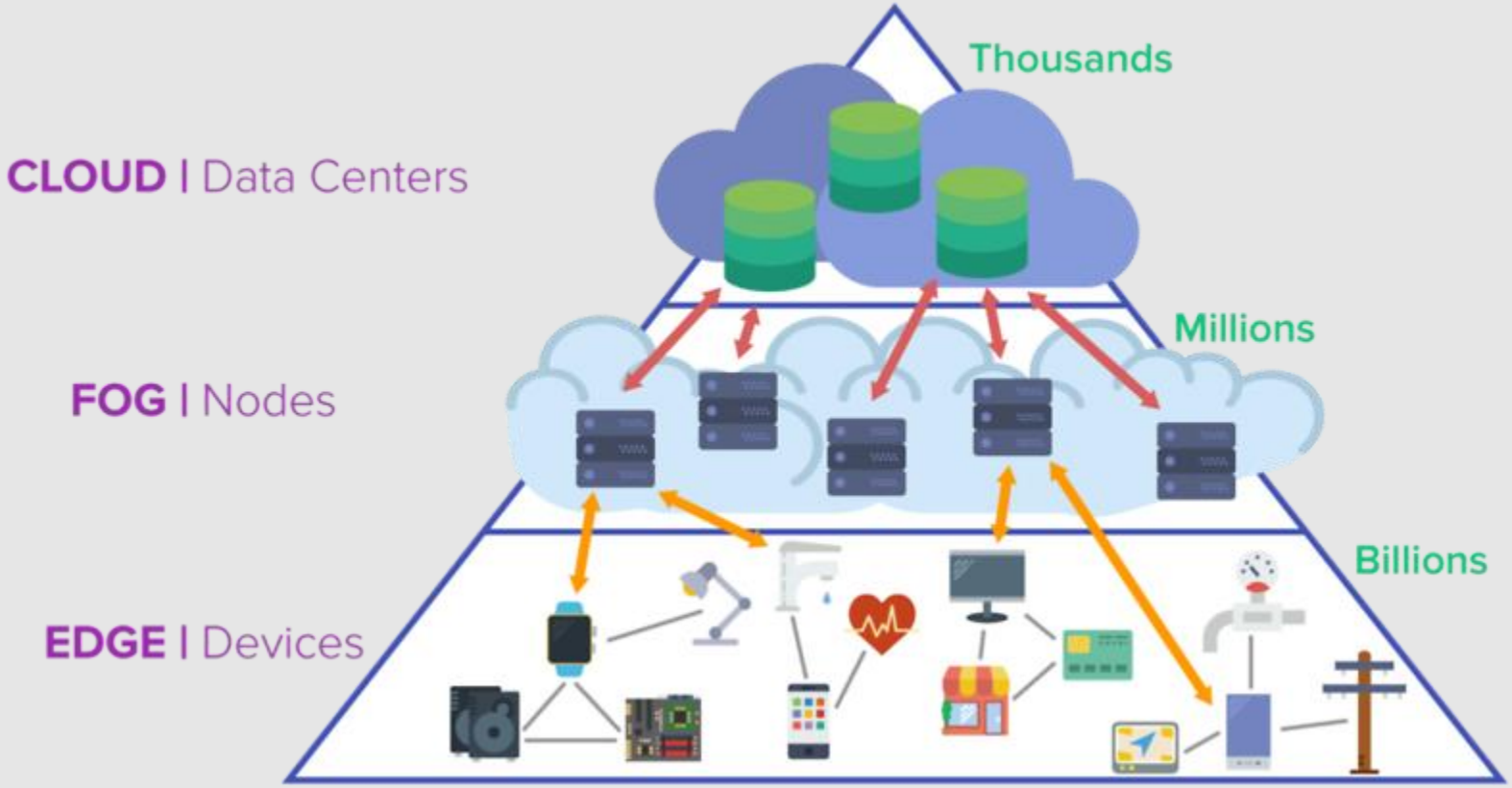
Azure
Machine Learning

Machine Learning

Typical E2E Process

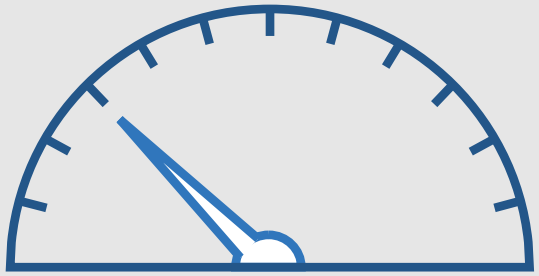


What is the Edge?

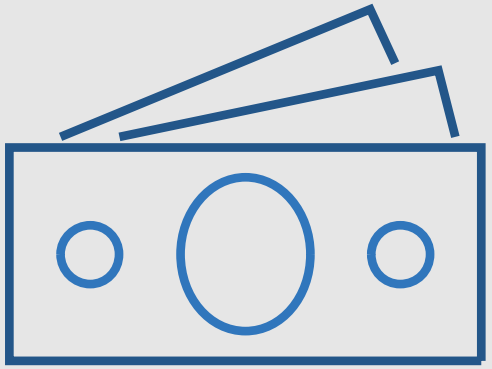


Imagimob AB

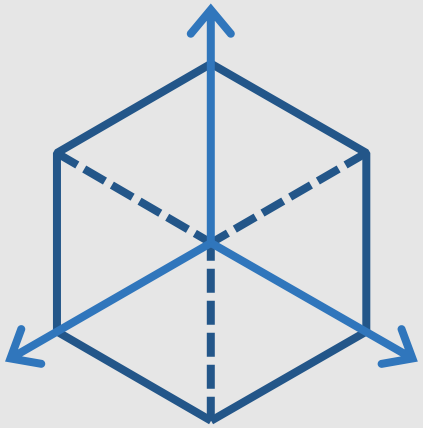
AI on the edge



Low latency



Scalability



Flexibility

ONNX as an intermediary format

- **Convert to Tensorflow for Android**
  - Convert a PyTorch model to Tensorflow using ONNX
- **Convert to CoreML for iOS**
  - https://github.com/onnx/onnx-coreml
- **Fine-tuning an ONNX model with MXNet/Gluon**
  - https://mxnet.apache.org/versions/1.3.1/tutorials/onnx/fine_tuning_gluon.html

ONNX Runtime

- High performance runtime for ONNX models
- Supports full ONNX-ML spec
- Extensible architecture to plug-in hardware accelerators
- API Support

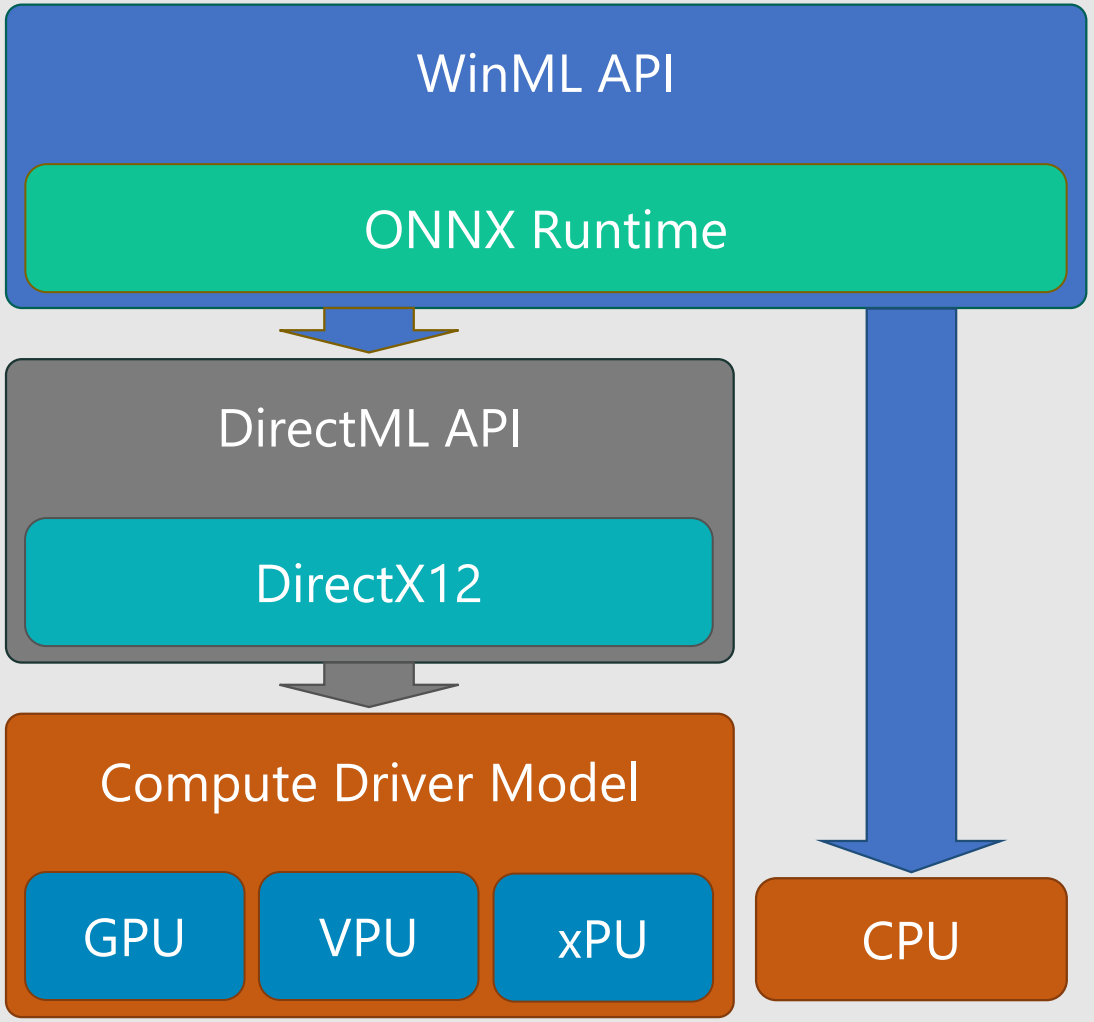
ONNX Runtime

Get Started Easily

Optimize Inferencing | Optimize Training (Preview)

OS	Windows	**Linux**	Mac	Android (Preview)	iOS (Preview)	
API	Python (3.5-3.7)	C++	**C#**	C	Java	Javascript (Node.js)	WinRT
Architecture	**X64**	X86	ARM64	ARM32		
Hardware Acceleration	Default CPU	ACL (Preview)	ArmNN (Preview)	**CUDA**	DirectML	
	DNNL	MKL-ML	MIGraphX (Preview)	NNAPI (Preview)		
	NUPHAR (Preview)	OpenVINO	Rockchip NPU (Preview)	TensorRT	Vitis AI (Preview)	
Installation Instructions	Install Nuget package Microsoft.ML.OnnxRuntime.Gpu					

Windows AI platform



- WinML
  - **Practical**, simple model-based API for ML inferencing on Windows
- DirectML
  - **Realtime, high control** ML operator API; part of DirectX family
- Compute Driver Model
  - Robust **hardware reach**/abstraction layer for compute and graphics silicon

Demo

https://github.com/rondagdag/onnx-pected/tree/master/GenerateONNX-AutoML

ONNX Docker Image

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

- getting started quickly with ONNX models
- ONNX converters
- inference using ONNX Runtime.

ONNX Runtime – Node JS

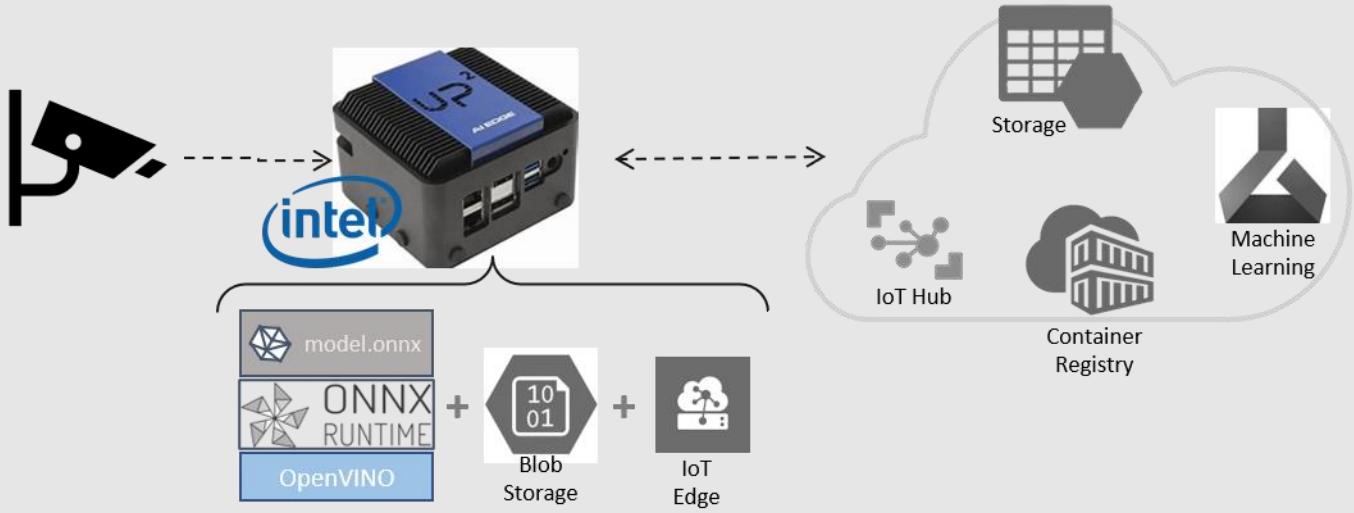
```
const ort = require('onnxruntime');

// use an async context to call onnxruntime functions.

async function main() {

        const session = await ort.InferenceSession.create('./model.onnx’);


        // prepare inputs. a tensor need its corresponding TypedArray as data

        const dataA = Float32Array.from([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]);

        const tensorA = new ort.Tensor('float32', dataA, [3, 4]);

        const tensorB = new ort.Tensor('float32', dataB, [4, 3]);



        const feeds = { a: tensorA, b: tensorB };    // prepare feeds. use model input names as keys.

        const results = await session.run(feeds);    // feed inputs and run

        const dataC = results.c.data;                // read from results

        console.log(`data of result tensor 'c': ${dataC}`);

}

main();
```

```
// create a new session and load the specific model.
       //
       // the model in this example contains a single MatMul node
       // it has 2 inputs: 'a'(float32, 3x4) and 'b'(float32, 4x3)
       // it has 1 output: 'c'(float32, 3x3)
```

ONNX

Reference implementation to use ONNX Runtime with Azure IoT Edge

- https://github.com/Azure-Samples/onnxruntime-iot-edge



ONNX.js

- ONNX.js is a JavaScript library for running ONNX models on browsers and on Node.js.
- ONNX.js has adopted Web Assembly and WebGL technologies
- optimized ONNX model inference runtime for both CPUs and GPUs.

https://github.com/microsoft/onnxjs

ONNX

ONNX.js

Compatibility

Desktop Platforms

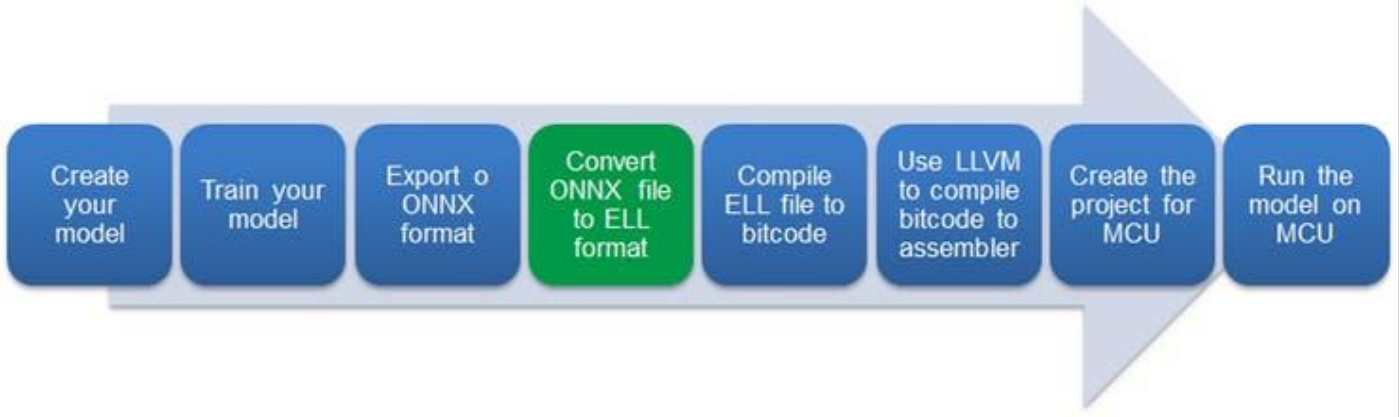
OS/Browser	Chrome	Edge	FireFox	Safari	Opera	Electron	Node.js
Windows 10	✔	✔	✔	-	✔	✔	✔
macOS	✔	-	✔	✔	✔	✔	✔
Ubuntu LTS 18.04	✔	-	✔	-	✔	✔	✔

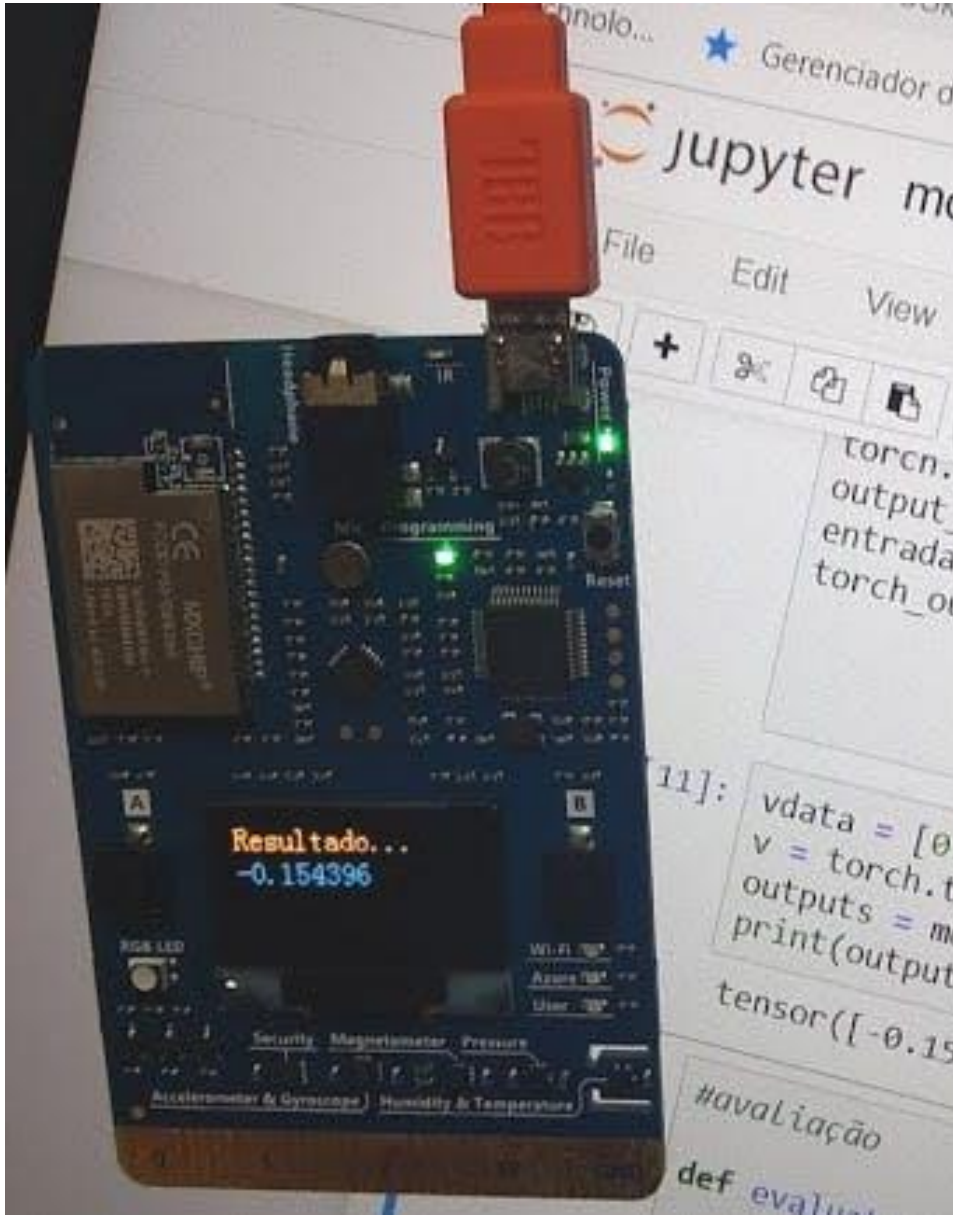
Mobile Platforms

OS/Browser	Chrome	Edge	FireFox	Safari	Opera
iOS	✔	✔	✔	✔	✔
Android	✔	✔	Coming soon	-	✔

Wait… there’s more

- Embedded Learning Library
  - https://github.com/microsoft/ELL
- Machine Learning Model Running on Azure IoT Starter Kit
  - https://www.hackster.io/waltercoan/machine-learning-model-running-on-azure-iot-starter-kit-f9608b





When to use ONNX?

- High Inferencing latency for production use
- Trained in Python - deploy into a C#/Java/JavaScript app
- Model to run resource constraint device (e.g. IoT/edge devices)
- Model to run on different OS or Hardware
- Combine running models created from different frameworks
- Training takes too long (transformer models)

Recap

- ✓ What is ONNX

  ONNX is an open standard so you can use the right tools for the job and be confident your models will run efficiently on your target platforms

- ✓ How to create ONNX models

  ONNX models can be created from many frameworks

- ✓ How to deploy ONNX models

  ONNX models can be deployed with Windows ML, .NET/Javascript/Python and to the cloud with Azure ML and the high performance ONNX Runtime

Try it for yourself!

ONNX Runtime is available now!

```
pip install onnxruntime
pip install onnxruntime-gpu
```

Documentation and samples at aka.ms/onnxruntime

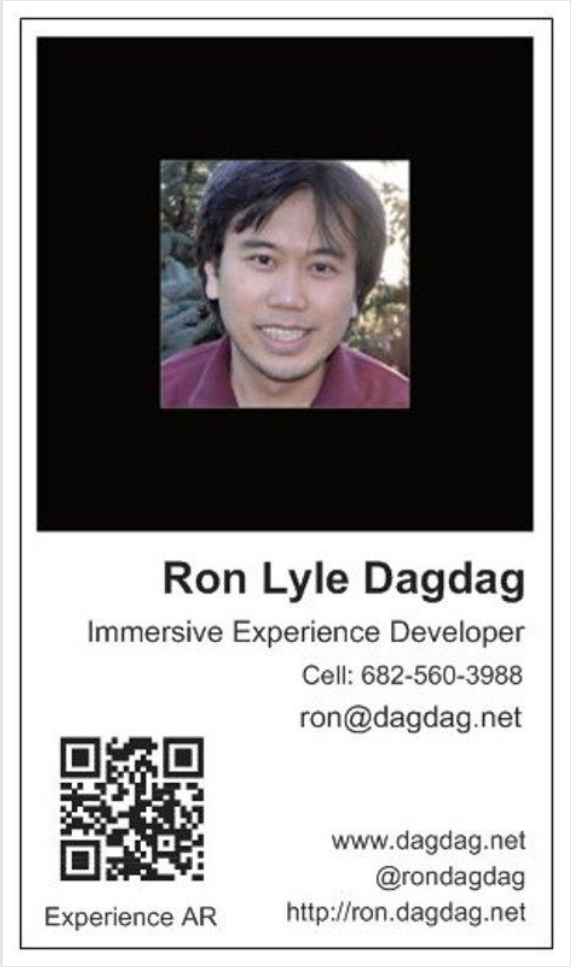
Source for Demo:

https://github.com/rondagdag/onnx-pected

http://bit.ly/ml-onnx

About Me

Ron Dagdag



- Lead Software Engineer at Spacee
- 4th year Microsoft MVP awardee
- Personal Projects
  www.dagdag.net
- Email: ron@dagdag.net
  Twitter @rondagdag
- Connect me via Linked In
  www.linkedin.com/in/rondagdag/
- Thanks for geeking out with me about ONNX