Ron Lyle Dagdag

Lead Software Engineer
@ Spacee

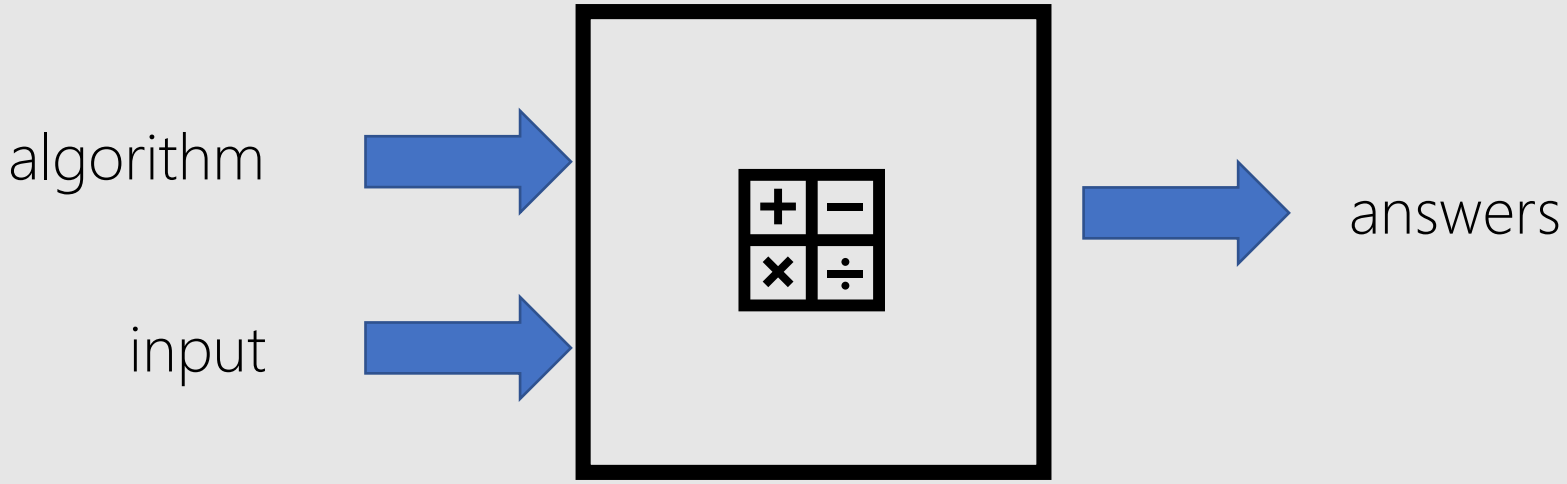
Leverage Power of Machine Learning with ONNX

Conf42: Machine Learning 2021

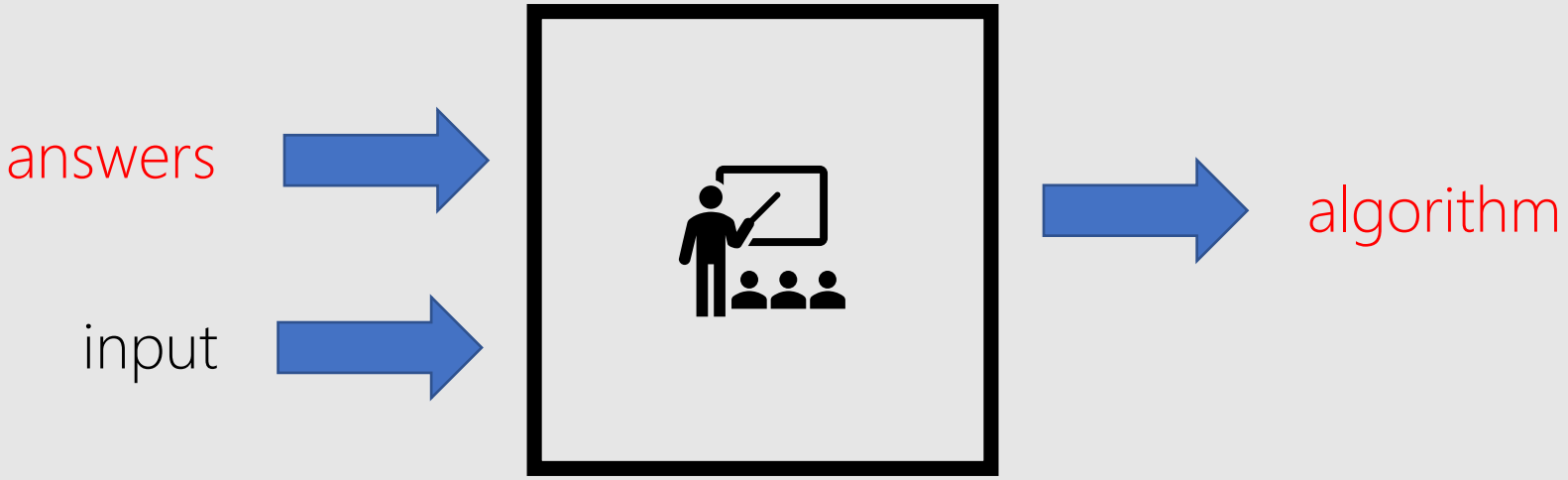
Thursday July 29 | 5PM GMT

ONNX
Not ONIX
Not ONYX

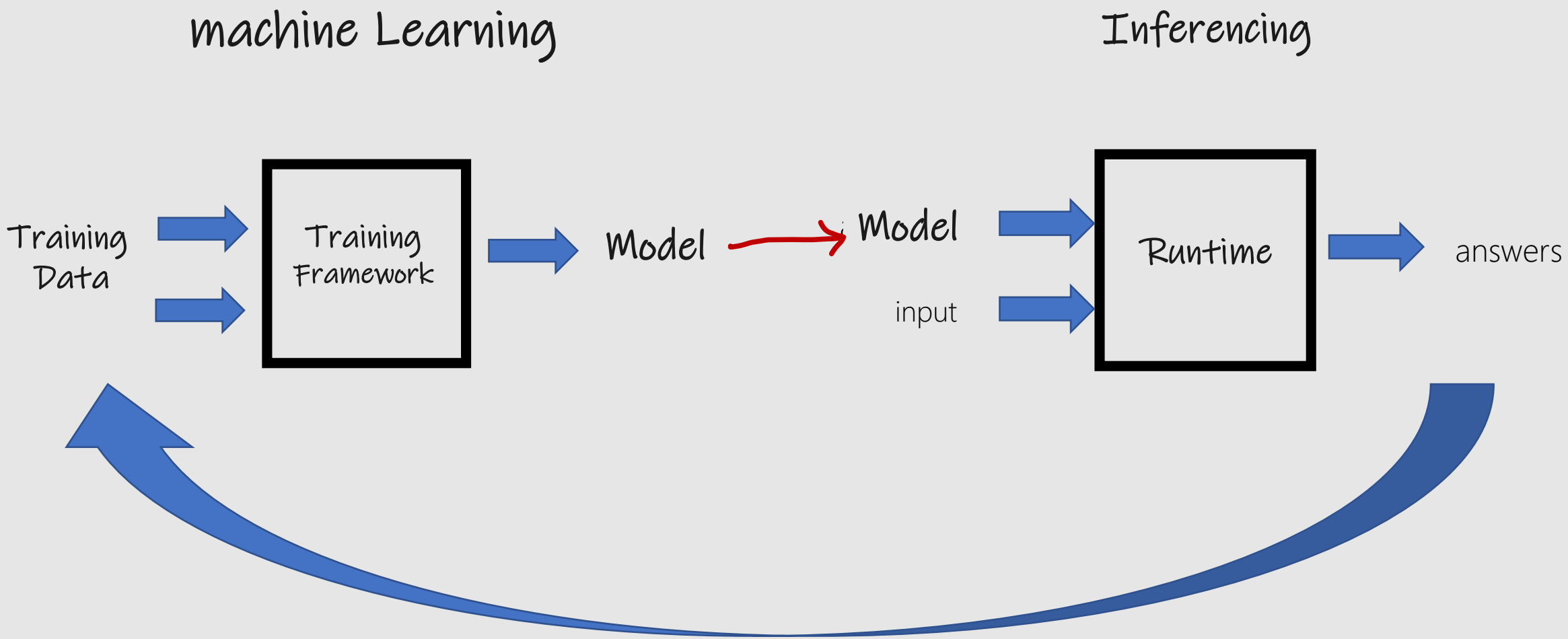
programming



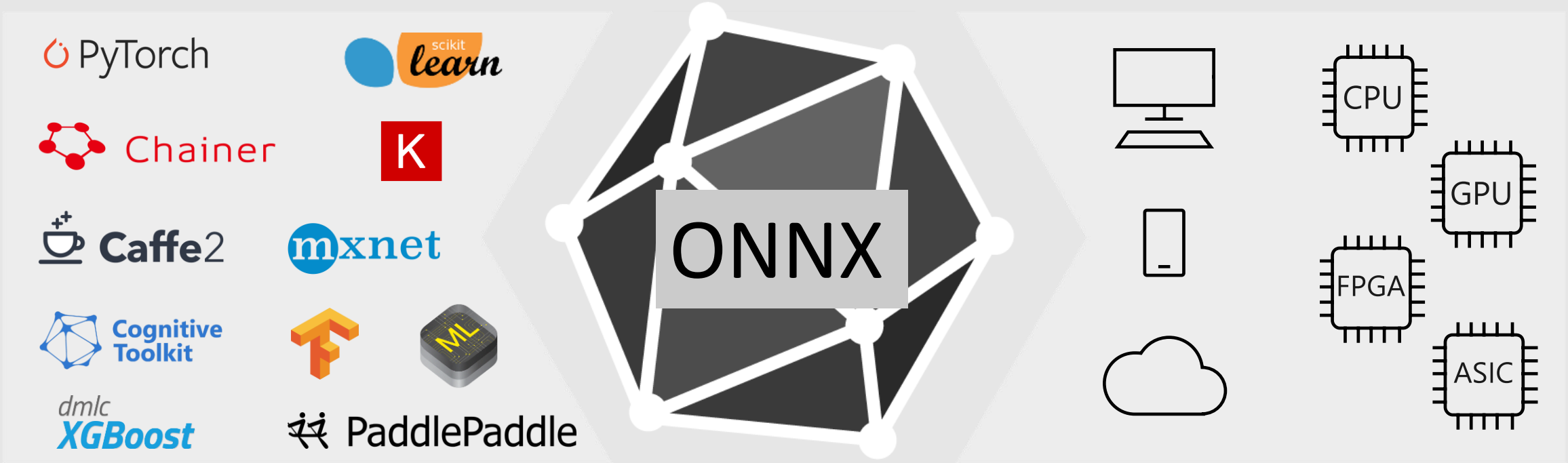
machine learning

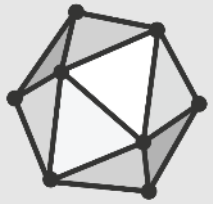


ML Primer


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
HAILO
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
vmware
WOLFRAM
Yandex
ZETANE

10.9k
Github Stars

2k Github
Forks

40+ Model
Zoo

197+
Contributors

1888+
Pull
Requests

LF AI
GRADUATE
PROJECT

When to use ONNX?

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

Agenda

- ✓ What is ONNX, When to use 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 · TensorFlow · Keras

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 VM

Windows Devices

IoT/Edge Devices

Other Devices
(iOS, Android, etc)

Frameworks

Caffe2 Chainer Cognitive Toolkit

mxnet PyTorch PaddlePaddle

ML.NET MathWorks dmlc XGBoost

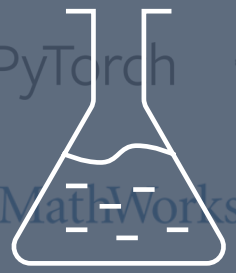
Services

Azure Custom Vision Service

Native support

Converters

Native support



Step 1: Create

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)

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

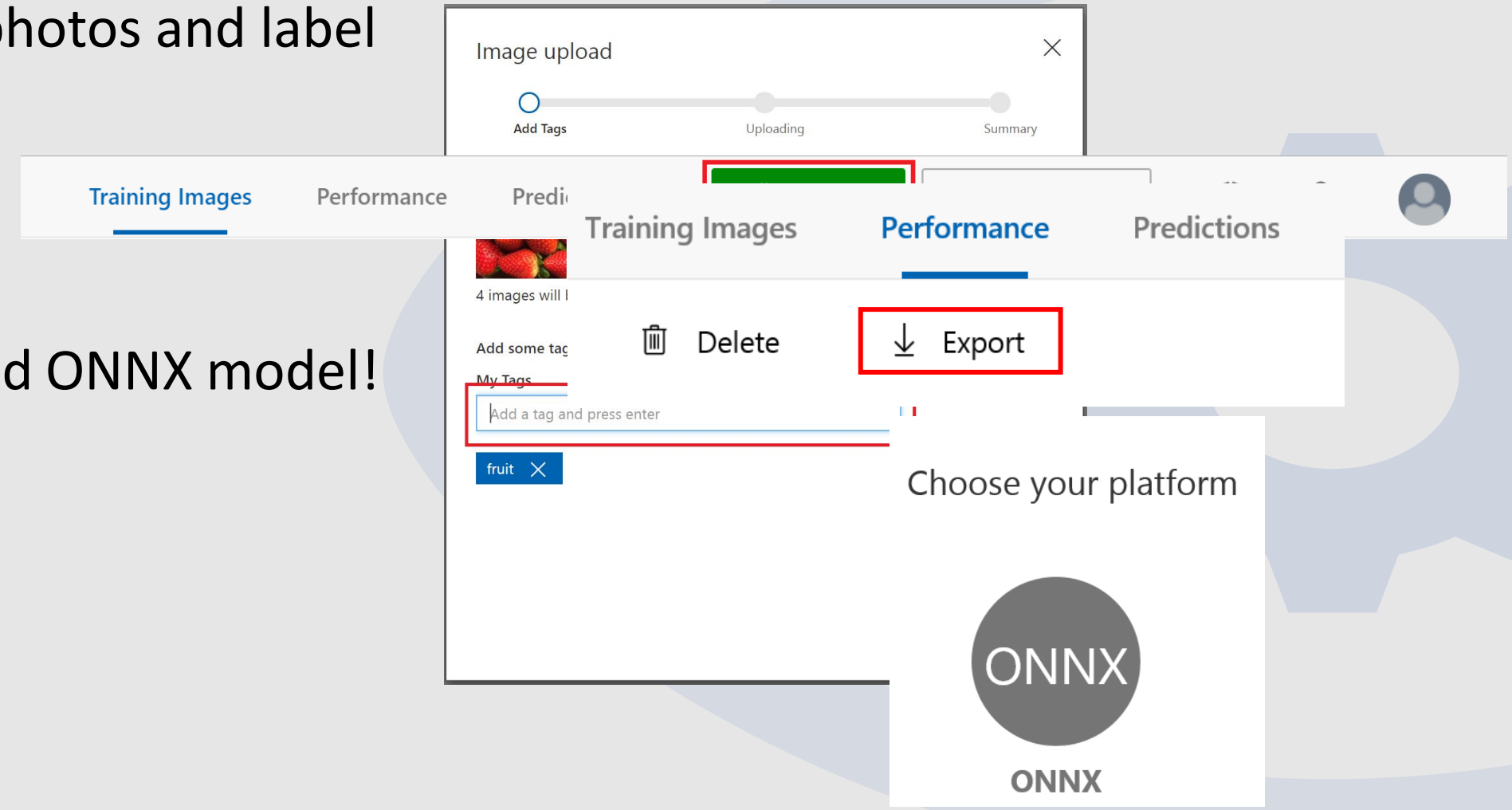
This collection of models take images as input, then classifies the major objects in the images into a set of predefined classes.

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

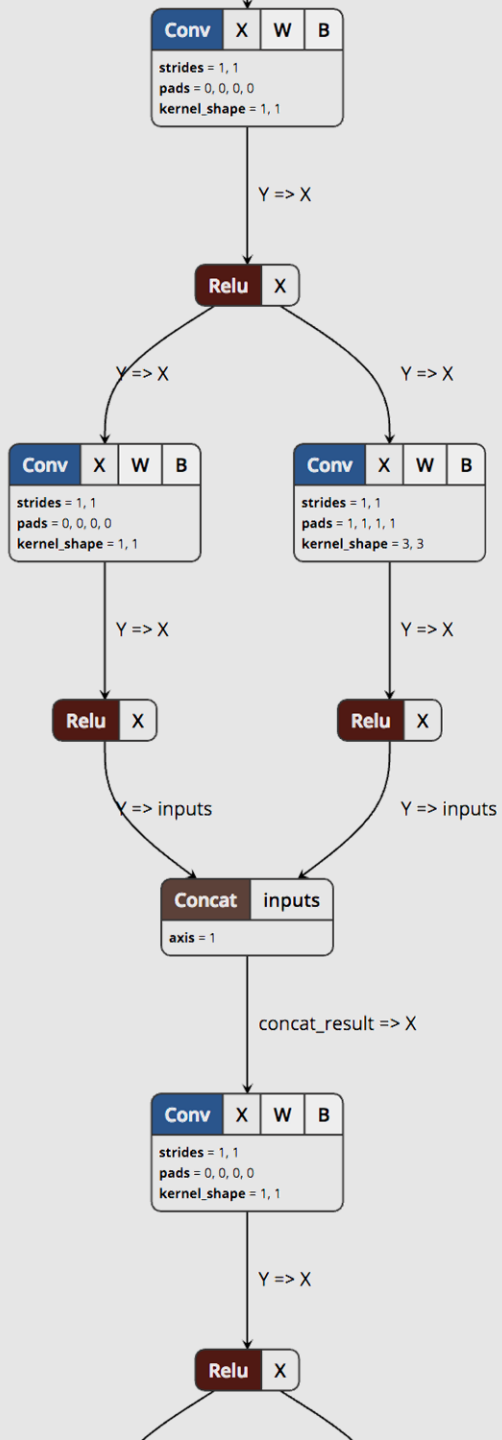
Convert models

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

https://github.com/onnx/tutorials

ONNX Models

Graph of operations



Netron

https://netron.app/

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

Convert models: PyTorch

```
import torch
import torch.onnx

model = torch.load(“model.pt”)

sample_input = torch.randn(1, 3, 224, 224)

torch.onnx.export(model, sample_input, “model.onnx")
```

Convert models: Keras

In [ ]:
```
import onnxmltools
from keras.models import load_model
```

In [ ]:
```
# Update the input name and path for your Keras model
input_keras_model = 'model.h5'

# Change this path to the output name and path for the ONNX model
output_onnx_model = 'model.onnx'
```

In [ ]:
```
# Load your Keras model
keras_model = load_model(input_keras_model)

# Convert the Keras model into ONNX
onnx_model = onnxmltools.convert_keras(keras_model)

# Save as protobuf
onnxmltools.utils.save_model(onnx_model, output_onnx_model)
```

Convert models:

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

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

Convert models:

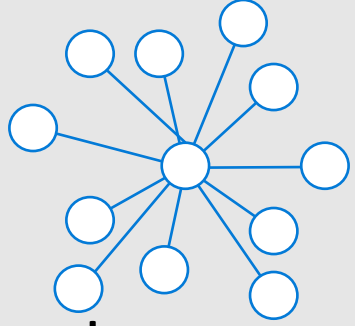

```
# Train a model.
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y)
clr = RandomForestClassifier()
clr.fit(X_train, y_train)

# Convert into ONNX format
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType
initial_type = [('float_input', FloatTensorType([None, 4]))]
onx = convert_sklearn(clr, initial_types=initial_type)
with open("rf_iris.onnx", "wb") as f:
    f.write(onx.SerializeToString())
```

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

https://github.com/onnx/tutorials

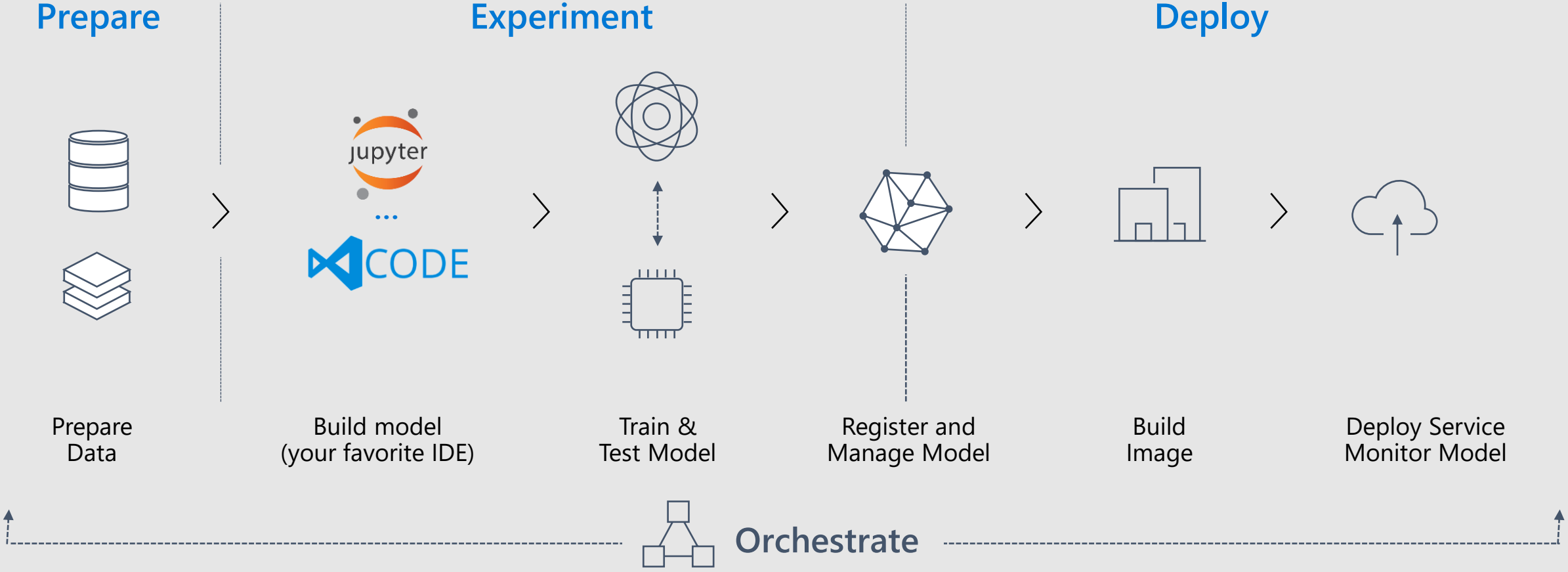
Train models in Azure Machine Learning



- 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



Frameworks

Caffe2 Chainer Cognitive Toolkit

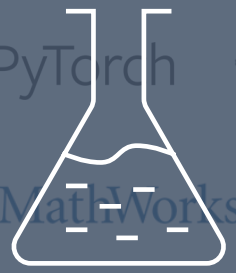
mxnet PyTorch PaddlePaddle

ML.NET MathWorks dmlc XGBoost

Native support

Converters

Native support



Step 1: Create

Services

Azure Custom Vision Service

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


Baker
vs
Starting a Bakery

Create

Frameworks

Caffe2 · Chainer · Cognitive Toolkit

mxnet · PyTorch · PaddlePaddle

ML.NET · MathWorks · dmlc XGBoost

ML · scikit learn · TensorFlow · Keras

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/Linux Devices

IoT Edge Devices

Other Devices
(iOS, etc)


Cloud
or
Edge

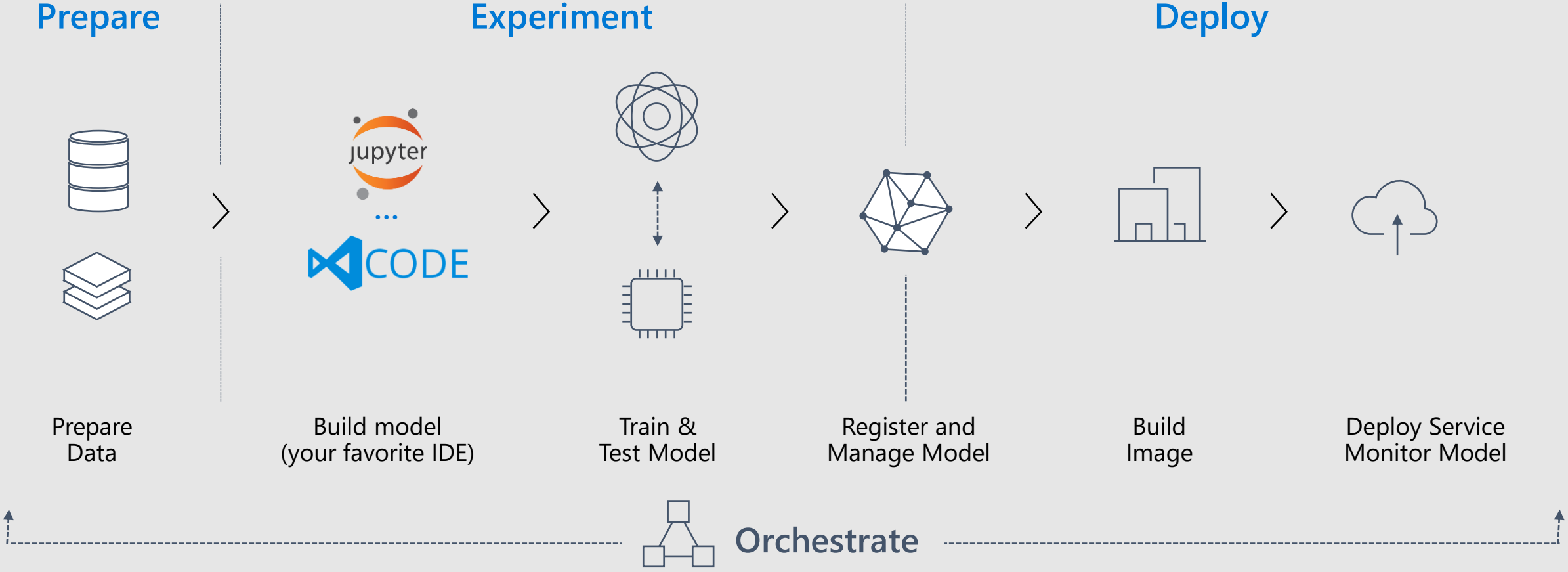
Deploy with Azure Machine Learning

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

Azure
Machine Learning

Machine Learning

Typical E2E Process



ONNX Docker Image

ONNX

onnx-base: Use published ONNX package from PyPi with minimal dependencies.

onnx-dev: Build ONNX from source with minimal dependencies.

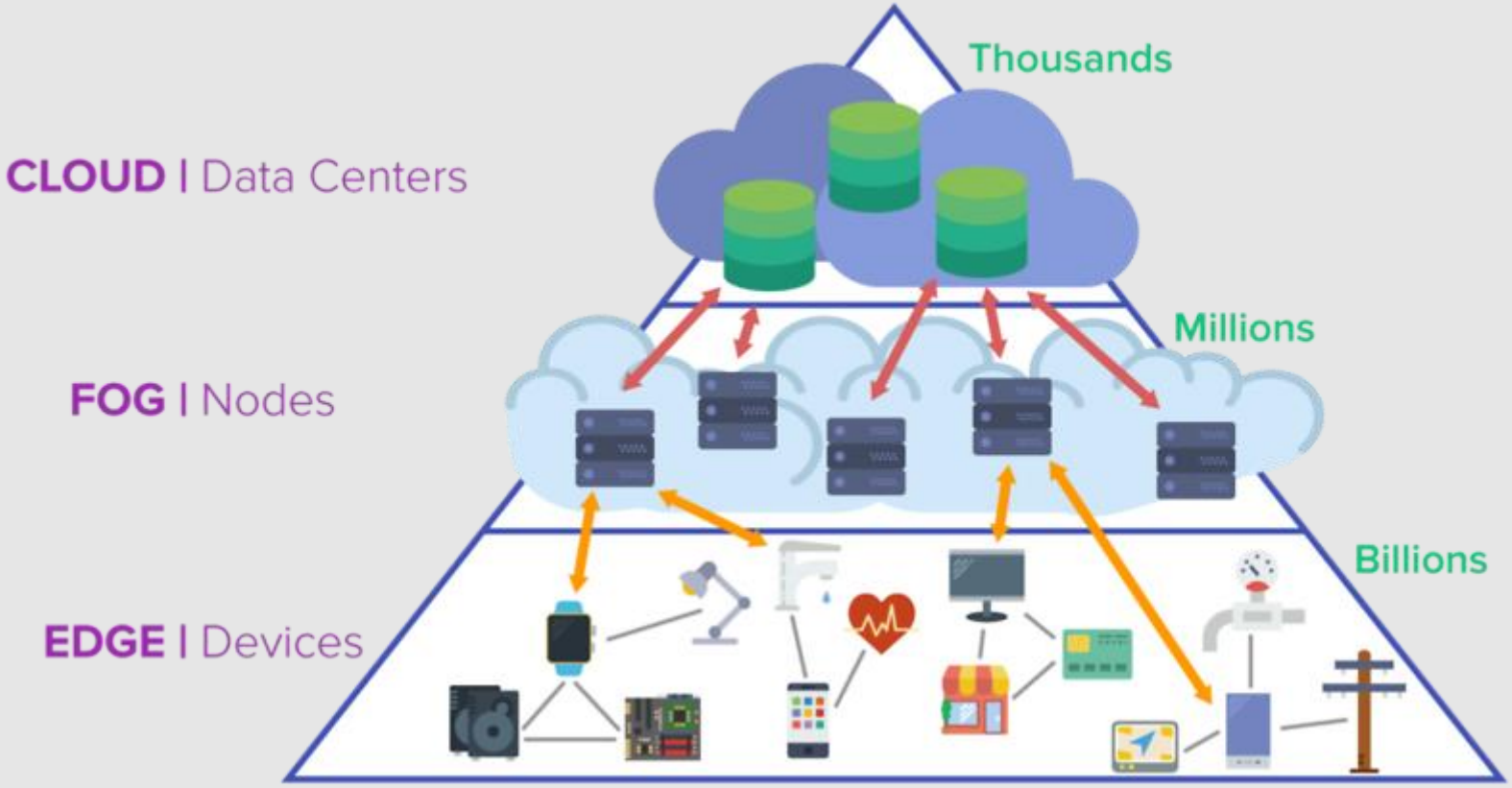
onnx-ecosystem: Jupyter notebook environment

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

Caffe2/PyTorch Docker

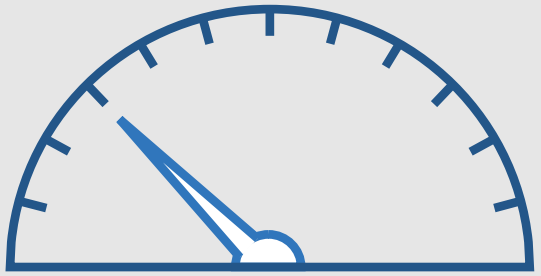
```
docker run -it --rm onnx/onnx-docker:cpu /bin/bash
```

What is the Edge?

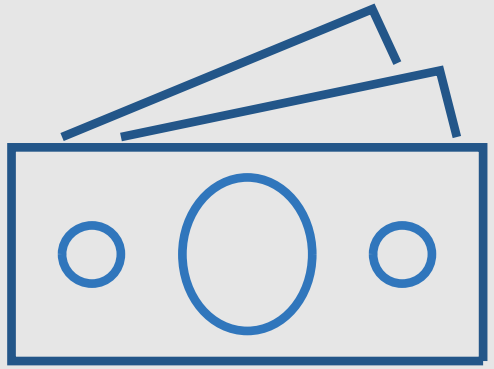


Imagimob AB

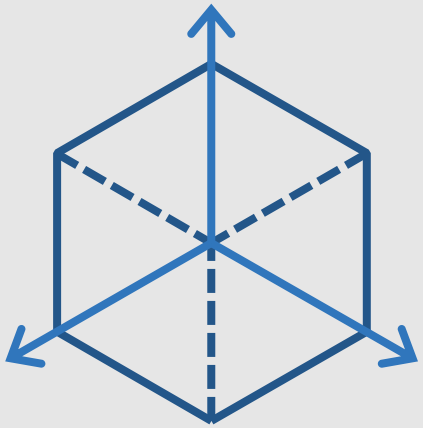
AI on the edge



Low latency



Scalability



Flexibility

ONNX Runtime

- High performance inference engine for ONNX models
- Founded and Open Sourced by Microsoft under MIT License
- Supports full ONNX-ML spec
- Extensible architecture to plug-in hardware accelerators
- Ships with Windows 10 as WinML
- onnxruntime.ai

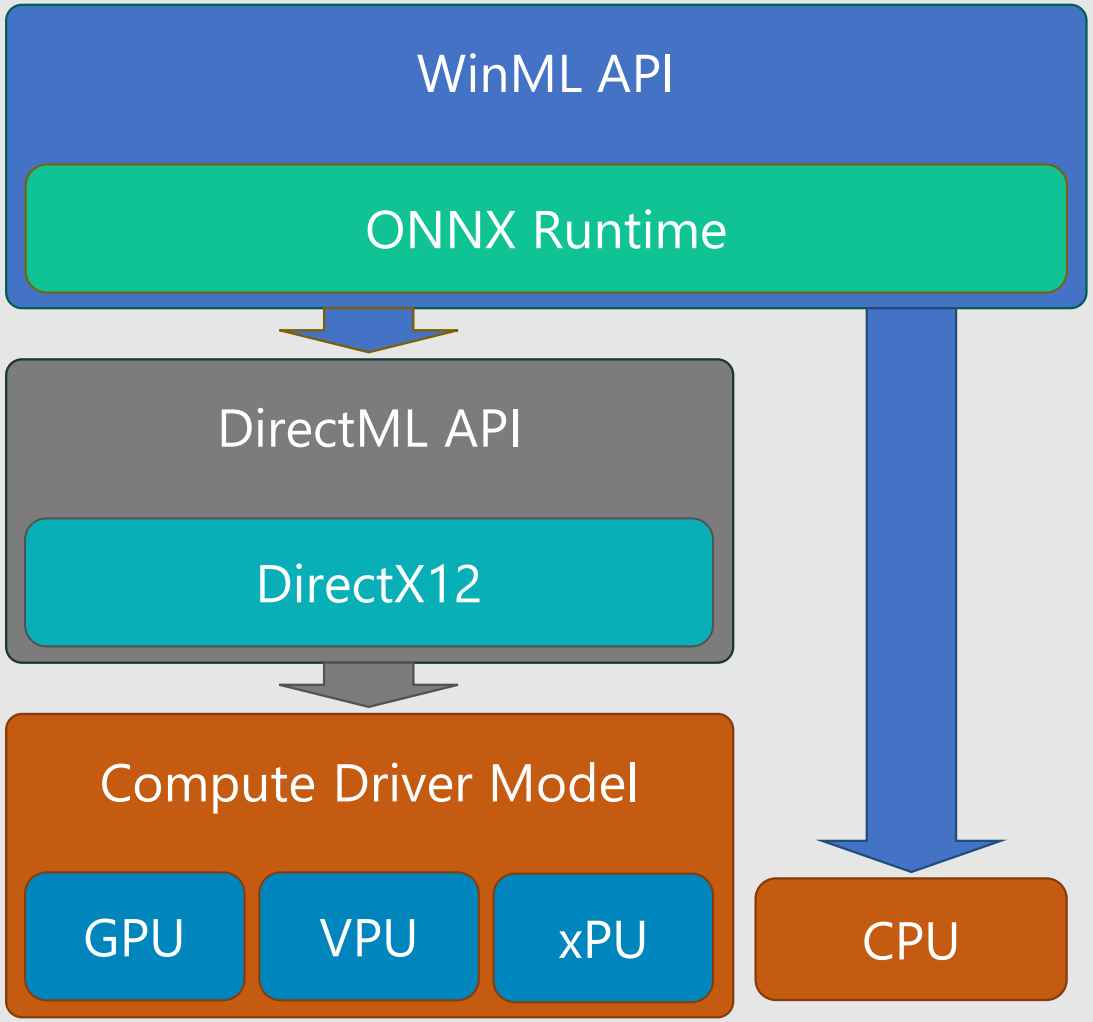
ONNX

ONNX Runtime

Optimize Inferencing	Optimize Training

Platform	Windows	**Linux**	Mac	Android	iOS	Web Browser (Preview)	
API	Python	C++	**C#**	C	Java	JS	Obj-C	WinRT
Architecture	**X64**	X86	ARM64	ARM32			
Hardware Acceleration	Default CPU	**CUDA**	DirectML	oneDNN	OpenVINO		
	TensorRT	NNAPI	ACL (Preview)	ArmNN (Preview)	CoreML (Preview)		
	MIGraphX (Preview)	NUPHAR (Preview)	Rockchip NPU (Preview)	Vitis AI (Preview)			
Installation Instructions	Install Nuget package **Microsoft.ML.OnnxRuntime.Gpu** Refer to **docs** for requirements.						

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

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

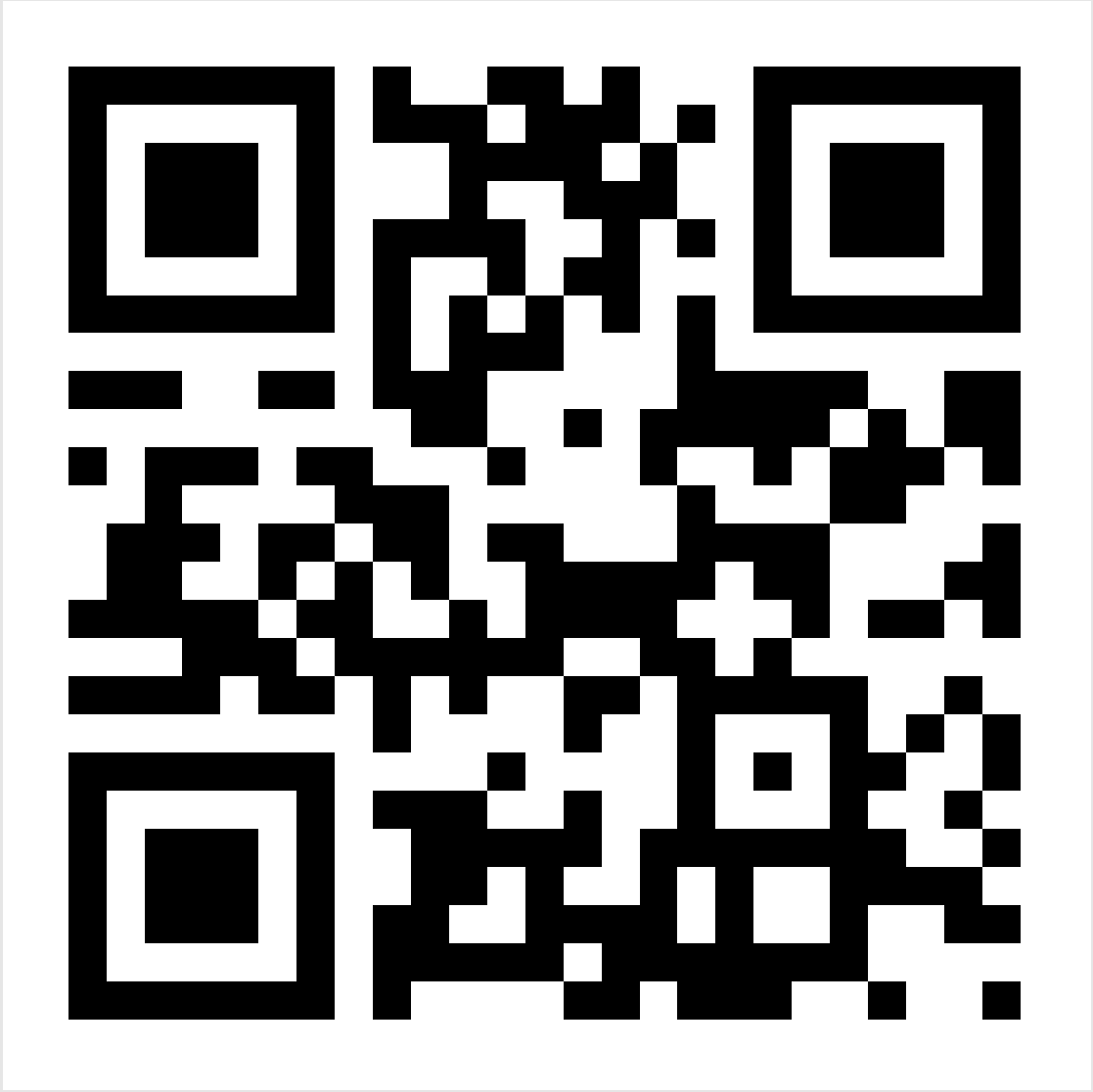
Compatibility

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	-	✓



http://bit.ly/ml-onnx

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

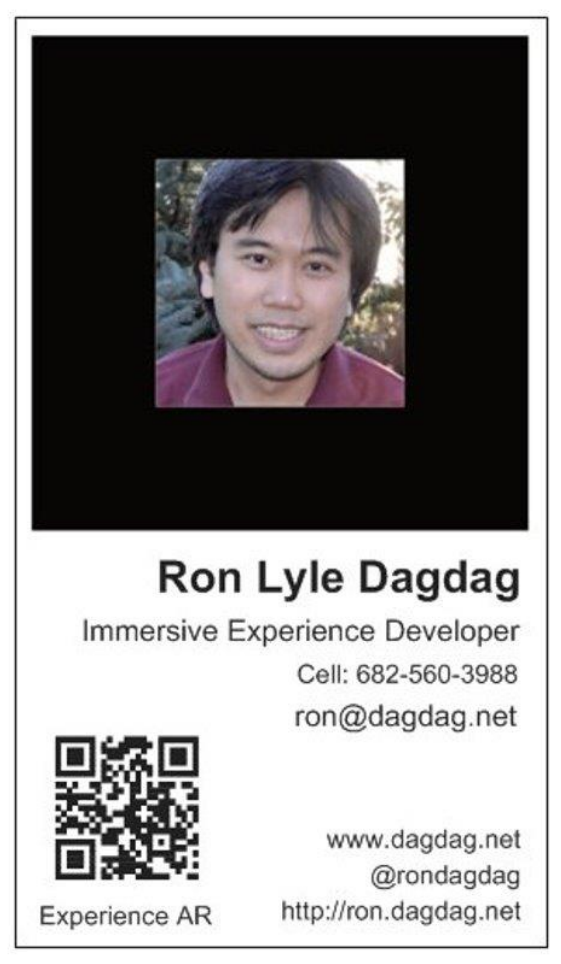
  ONNX models can be created from many frameworks

- ✓ 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

About Me

Ron Dagdag



- Lead Software Engineer at Spacee
- 5th 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

Hackster Portfolio

www.dagdag.net

@rondagdag