aka.ms/odsc-onnx

Please get started with the pre-requisite and getting started steps

OPEN DATA SCIENCE CONFERENCE

London | Nov. 19 - Nov. 22 2019

FROM RESEARCH TO PRODUCTION:

performant cross-platform ML/DNN model inferencing on cloud and edge with ONNX Runtime

FAITH XU | Senior Program Manager @ Microsoft

PRABHAT ROY | Data and Applied Scientist @ Microsoft

Agenda – What we’ll cover today

- **INTRODUCTION TO ONNX**
- **ACTIVITY A:** Train an image classification model in PyTorch and convert to ONNX format for inferencing
- **ACTIVITY B:** Train a PyTorch model and deploy for production usage

Why now?

Trends and Growth Areas

Research -> Industry

- Automated Machine Learning services
- Startups – applied AI
- Hosted services for cloud compute
- Hardware investments

Connectivity, compute, and resources

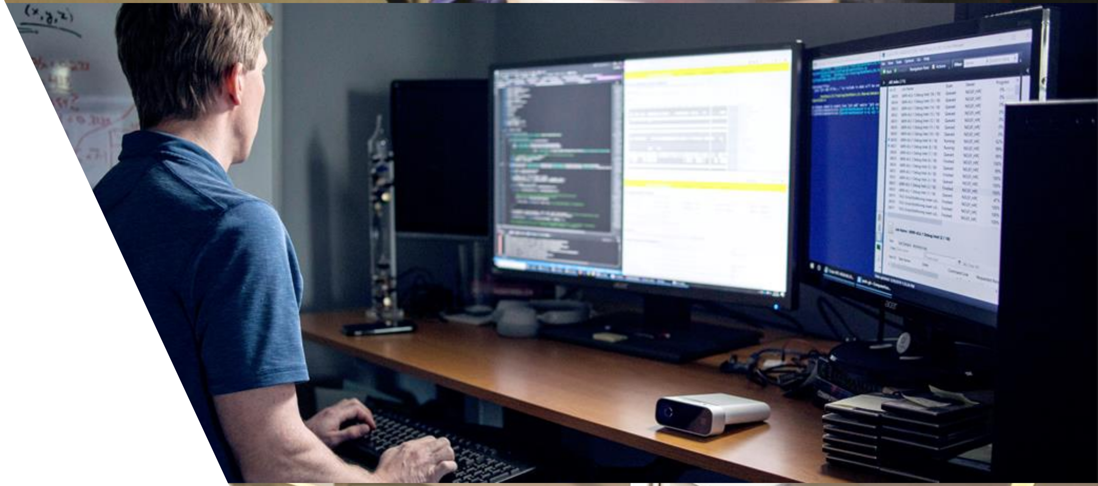
- Infinite storage and compute in the cloud
- CPU, GPUs for training
- LOTS of data

Application spans across all industries

- Healthcare, farming, gaming, manufacturing, consumer products, and more

Investments in AI education and jobs

- Universities
- ML Engineer



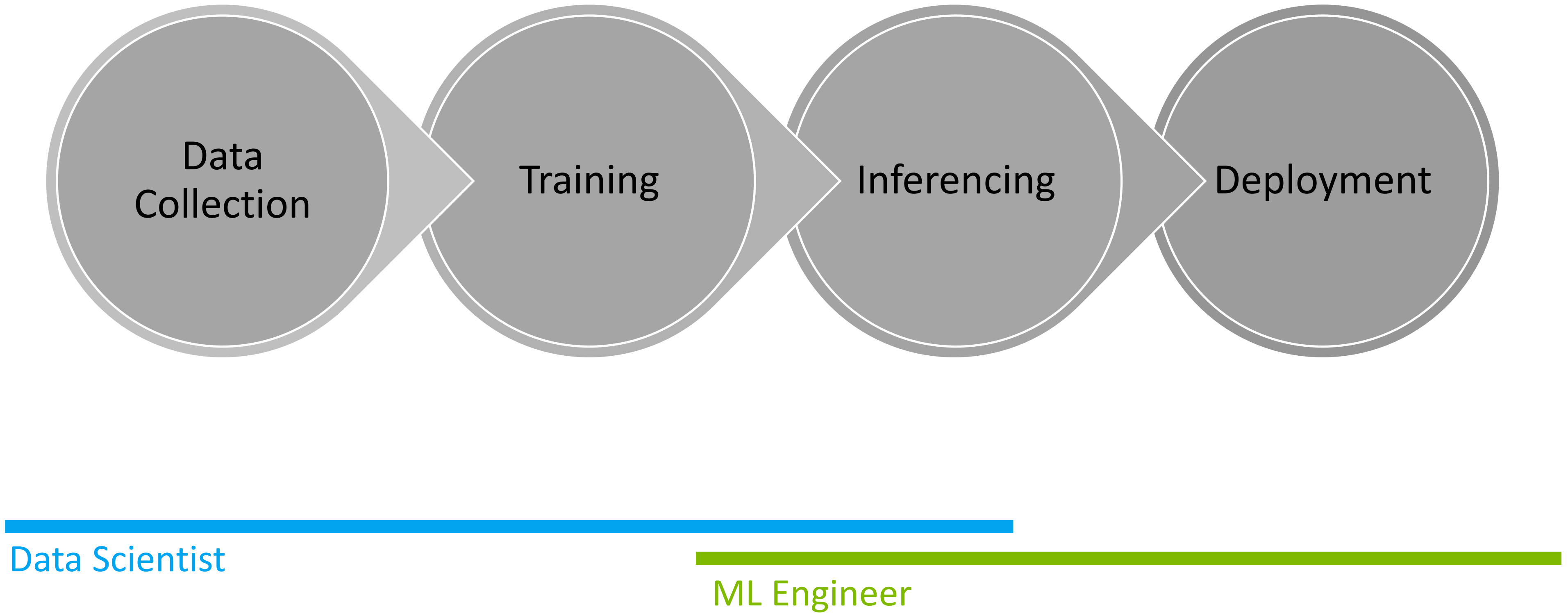
Product teams want to incorporate ML

Microsoft 365 | Windows | Microsoft Dynamics 365

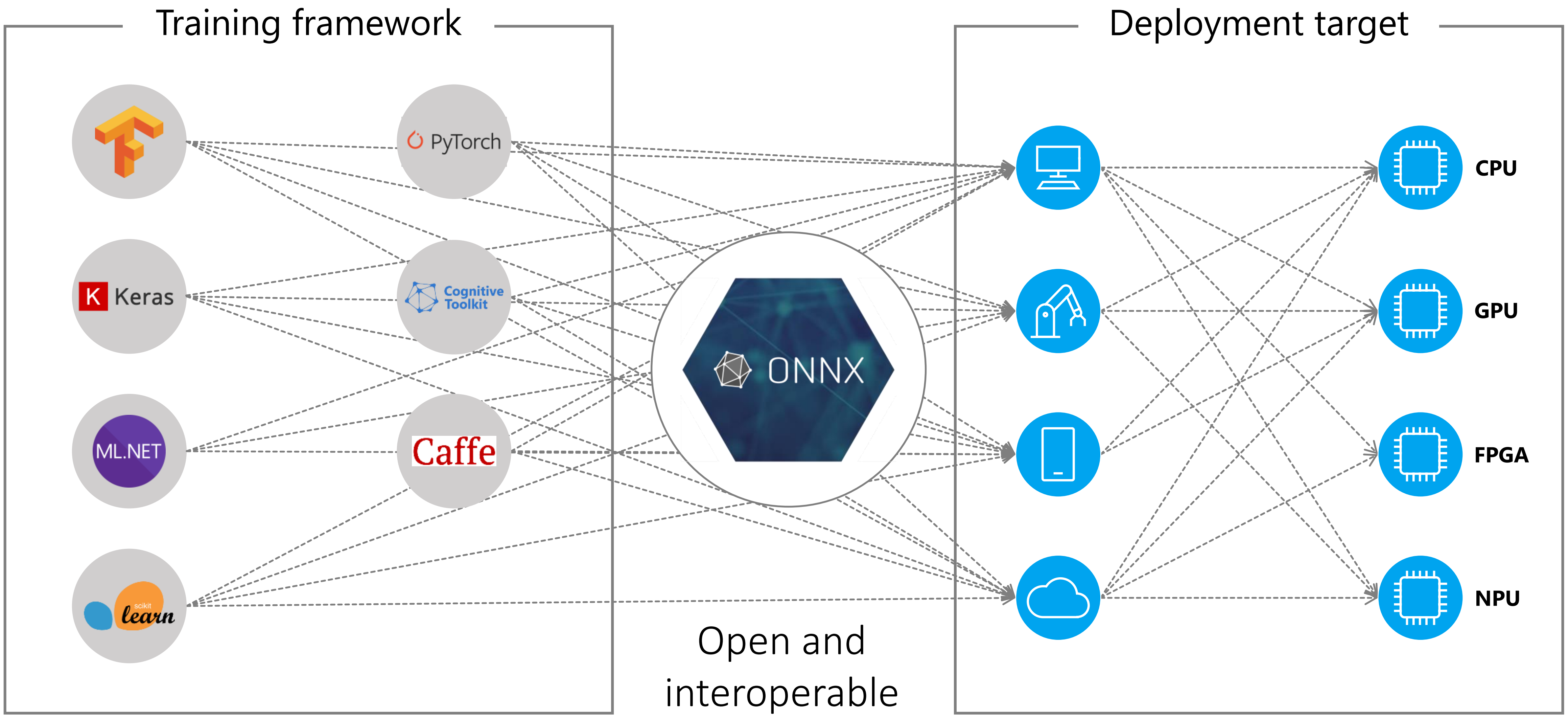
Skype | Bing | Microsoft HoloLens

Microsoft | Research | Office 365 | XBOX

ML Models: Research to Production



Reality



Open and interoperable

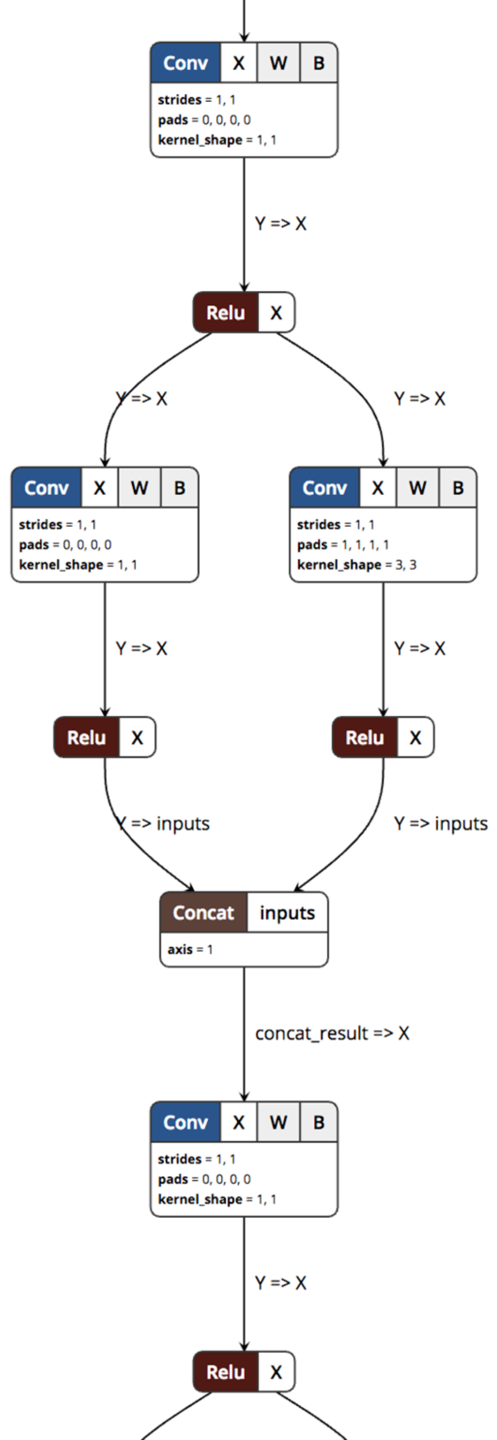
ONNX

https://github.com/onnx

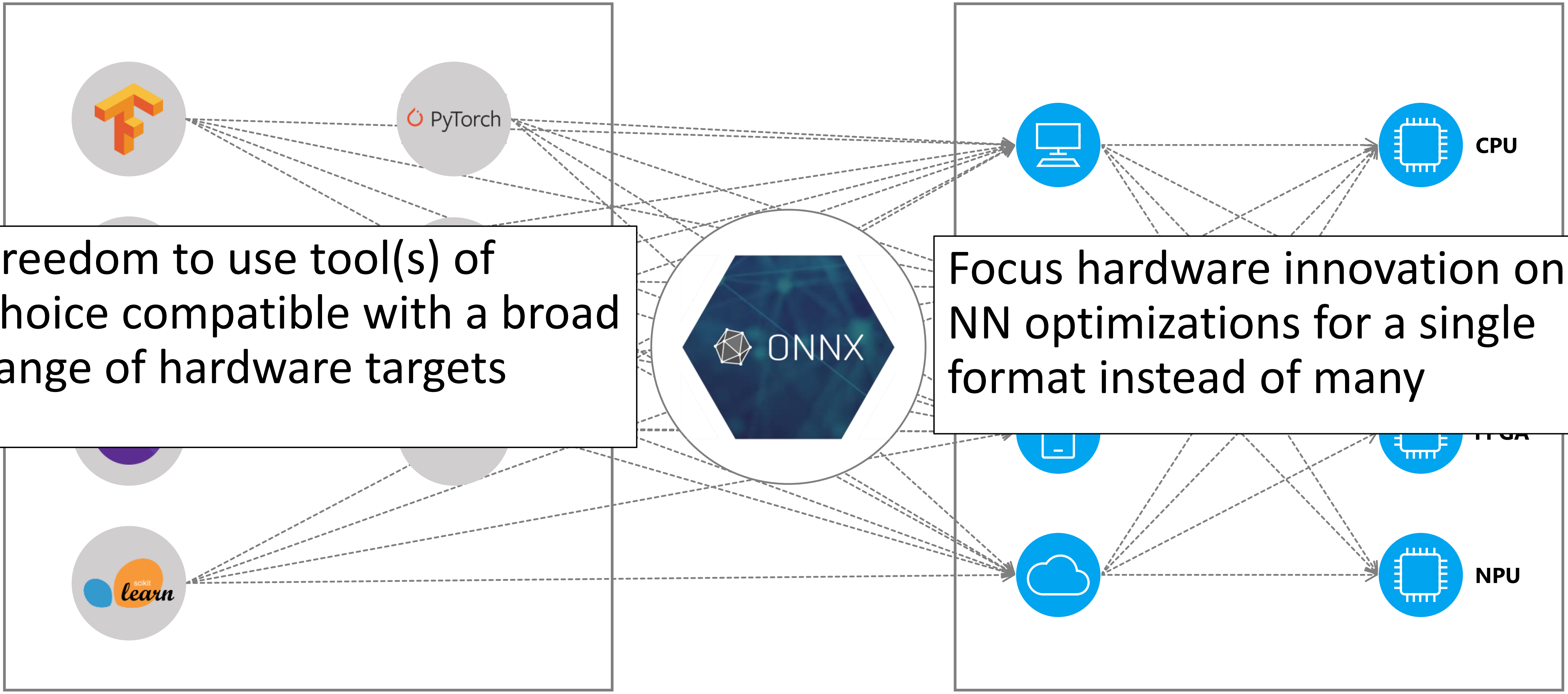
OPEN **N**EURAL **N**ETWORK E**X**CHANGE

What is ONNX?

- **Interoperable standard** format for AI models consisting of:
  - common Intermediate Representation (**IR**)
  - full operator **spec**
- Model = graph composed of computational nodes, based on Google protobuf
- Graph = Compact and cross-platform representation for serialization
- Supports both DNN and traditional ML
- Backward compatible with comprehensive versioning



What does this provide?



Freedom to use tool(s) of choice compatible with a broad range of hardware targets

Focus hardware innovation on NN optimizations for a single format instead of many

Framework Compatibility

Caffe2 mxnet PyTorch PaddlePaddle

Chainer ML.NET MathWorks dmlc XGBoost

Cognitive Toolkit ML scikit learn K

ONNX Community

LF AI

Alibaba Group 阿里巴巴集团 AMD arm aws Bai du 百度 BITMAIN

CEVA Facebook Open Source GRAPHCORE habana Hewlett Packard Enterprise HUAWEI

IBM IDEIN Idein Inc intel AI MathWorks MAXAR MEDIATEK Microsoft

Neural Network Libraries mi NVIDIA NXP Oath: A Verizon company OctoML

Preferred Networks Qualcomm sas skymizer SYNOPSYS Tencent unity

Open Governance

Steering Committee

Prasanth Pulavarthi (Microsoft)

Joe Spisak (Facebook)

Vin Sharma (Amazon)

Harry Kim (Intel)

Dilip Sequeira (NVIDIA)

SIG (special interest group)

Architecture/Infrastructure

Lu Fang (Facebook)
Ke Zhang (Microsoft)

Operators

Michał Karzyński (Intel)
Emad Barsoum (Microsoft)

Converters

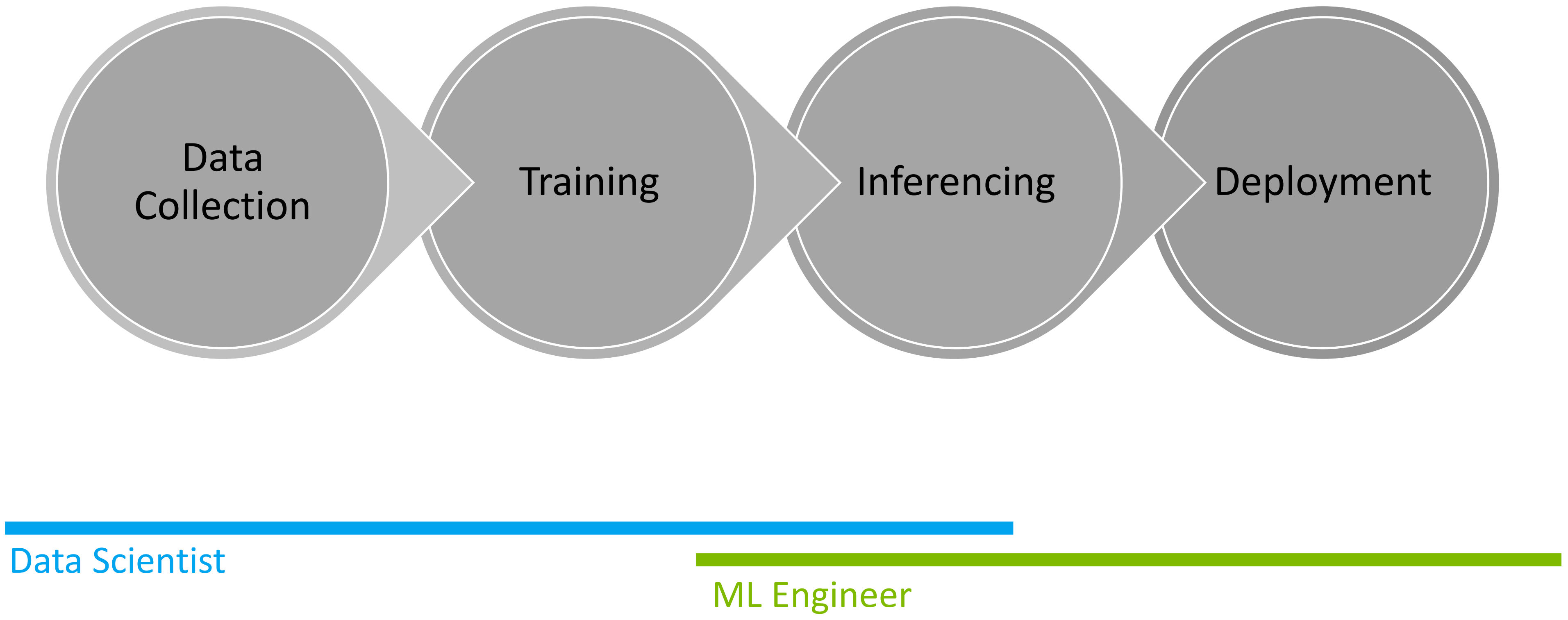
Chin Huang (IBM)
Guenther Schmuelling (Microsoft)

Working Groups

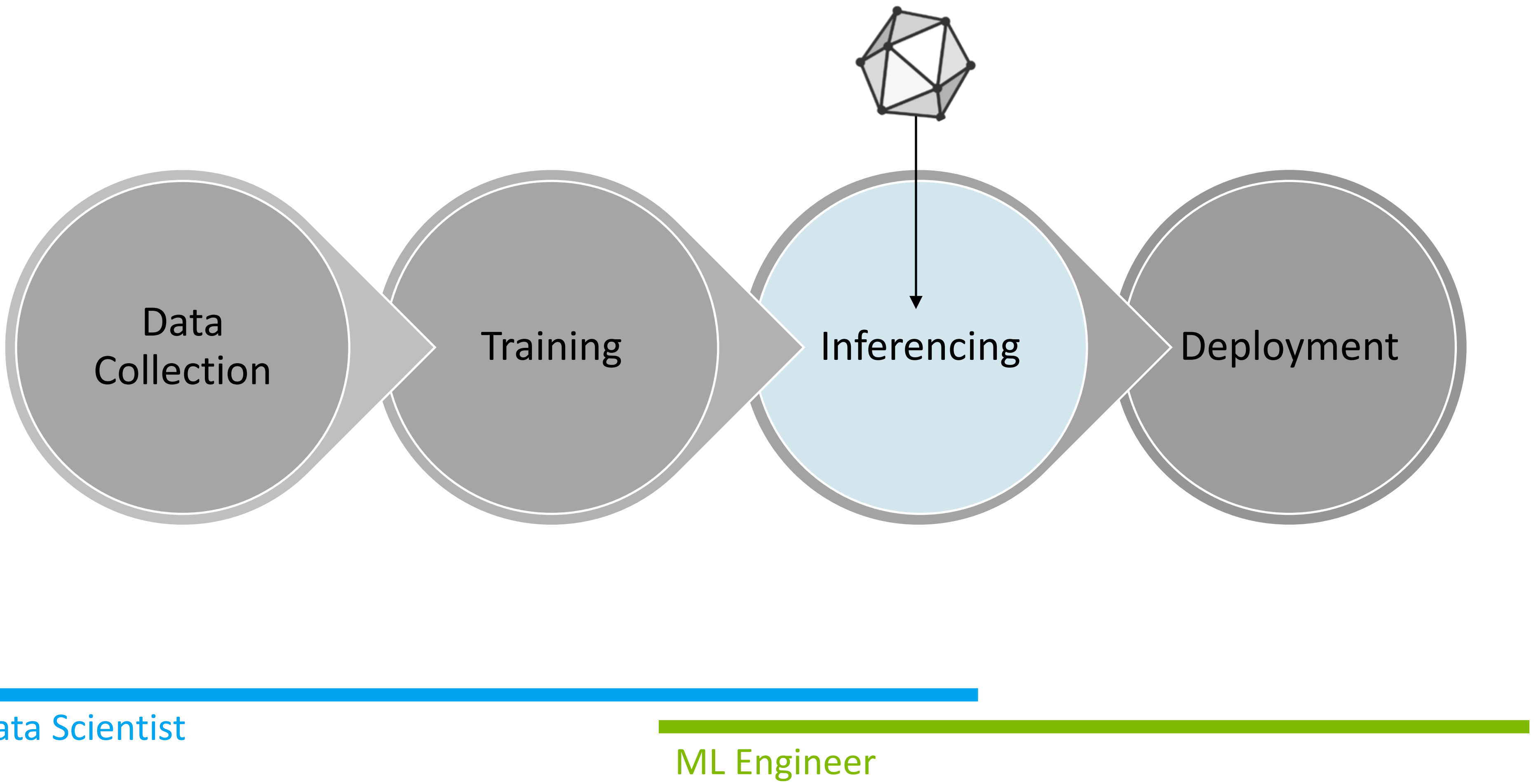
Training

Edge/Mobile

ML Models: Research to Production



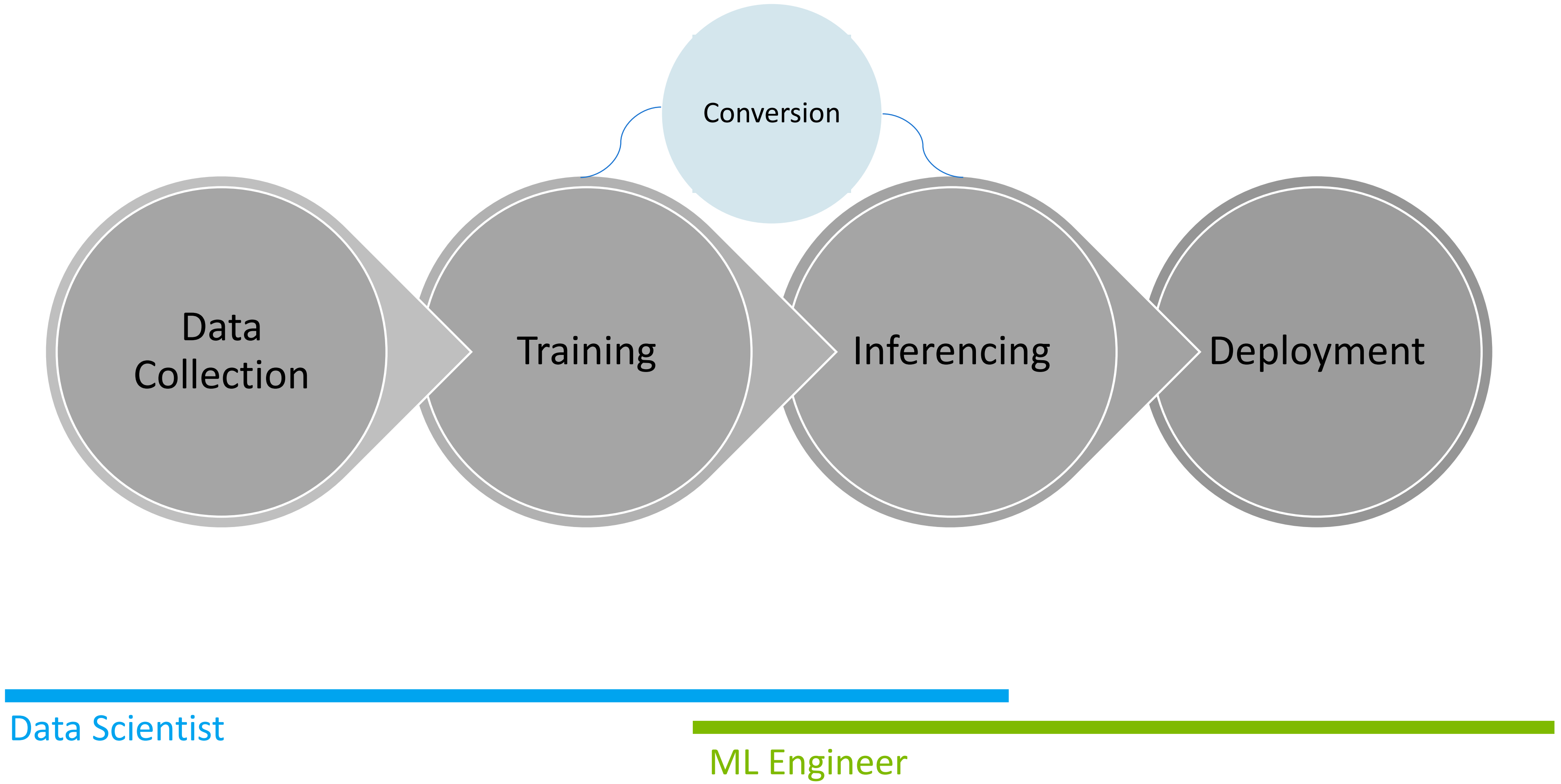
ML Models: Research to Production



How do I get an ONNX model?

- Get a pre-trained ready to use model from the ONNX Model Zoo
- Use a model builder service that supports export to the ONNX format
- **Convert an existing model from another framework**

ML Models: Research to Production



Open Source converters for popular frameworks

Tensorflow: onnx/tensorflow-onnx

PyTorch (native export)

Keras: onnx/keras-onnx

Scikit-learn: onnx/sklearn-onnx

CoreML: onnx/onnxmltools

LightGBM: onnx/onnxmltools

LibSVM: onnx/onnxmltools

XGBoost: onnx/onnxmltools

SparkML (alpha): onnx/onnxmltools

CNTK (native export)

Examples: Model Conversion

```
from keras.models import load_model
import keras2onnx
import onnx

keras_model = load_model("model.h5")

onnx_model = keras2onnx.convert_keras(keras_model,
keras_model.name)

onnx.save_model(onnx_model, 'model.onnx')
```

```
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")
```

```
python -m tf2onnx.convert
         --input frozen_model.pb
         --inputs input_batch:0, lengths:0
         --outputs top_k:1
         --fold_const
         --opset 8
         --output deepcc.onnx
```

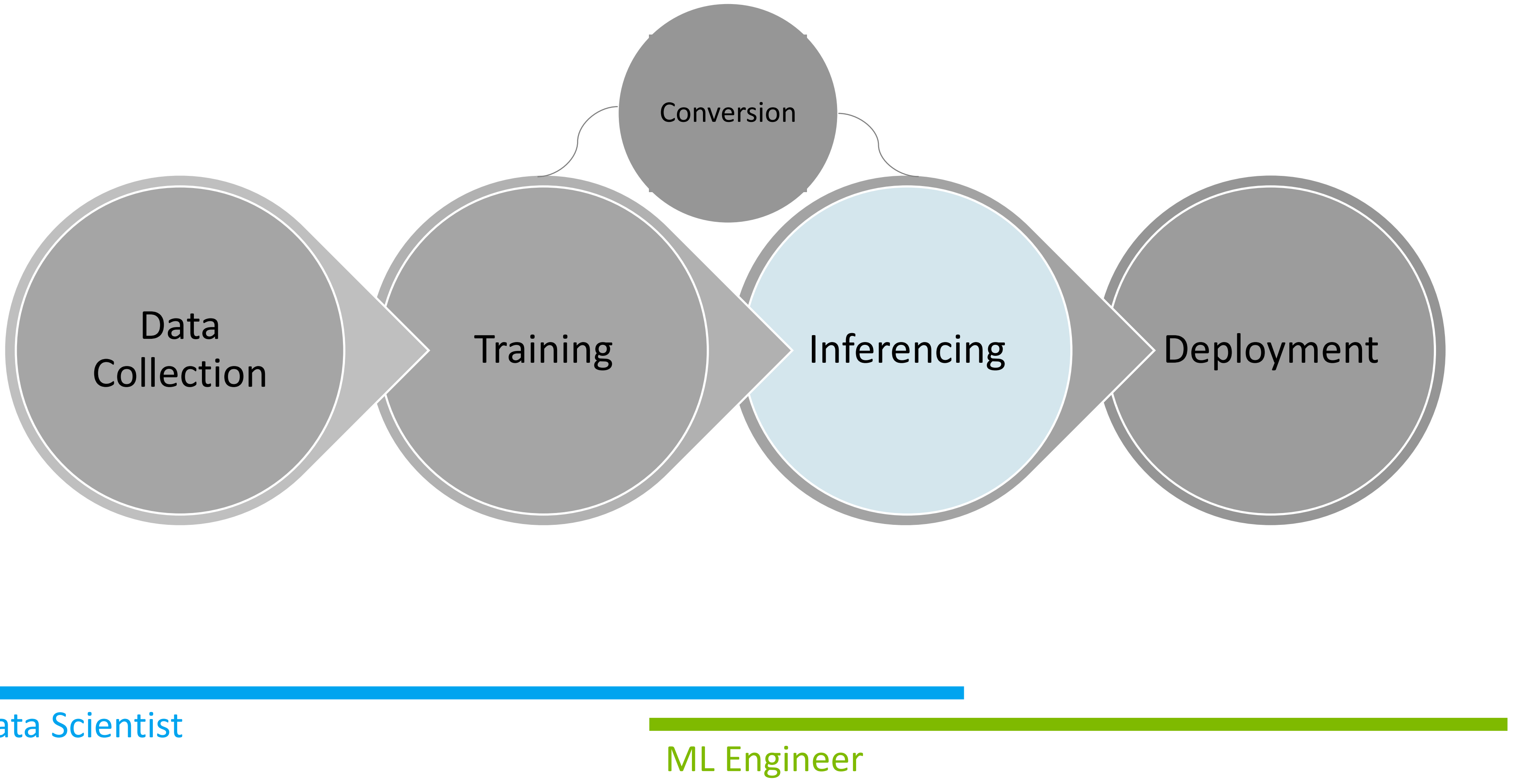
```
import numpy as np
import chainer
from chainer import serializers
import onnx_chainer

serializers.load_npz("my.model", model)

sample_input = np.zeros((1, 3, 224, 224), dtype=np.float32)
chainer.config.train = False

onnx_chainer.export(model, sample_input, filename="my.onnx")
```

ACTIVITY A

Train an image classification model in PyTorch and convert to ONNX format for inferencing

Inferencing ONNX models

ML Models: Research to Production



ONNX Runtime

aka.ms/onnxruntime

github.com/microsoft/onnxruntime

ONNX Runtime is an open source high performance **Inference Engine** for ONNX models

github.com/microsoft/onnxruntime

ONNX Runtime can run all operators defined in the ONNX spec

- ONNX domain (DNN) and ONNX-ML (traditional)
- Backwards and forwards compatible to minimize versioning issues with software or model upgrades
- Flexibility for custom operators not in the spec

Cross platform, multi language API

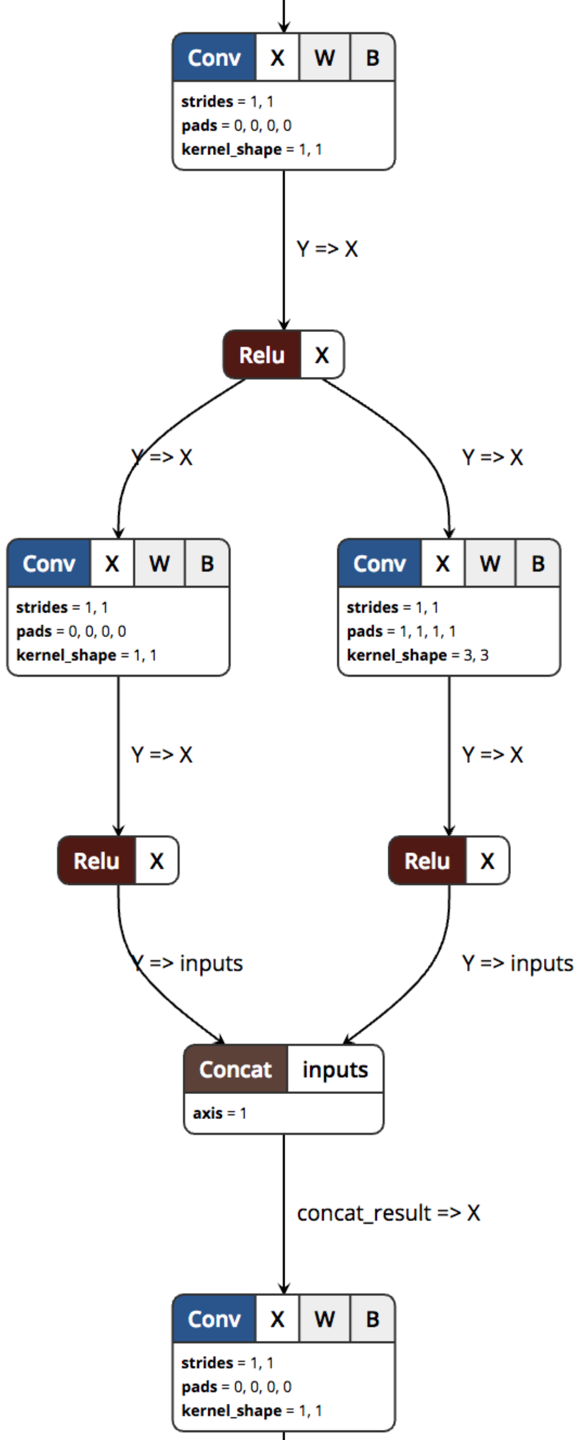
- Windows, Linux, Mac
- X64, X86, ARM
- CPU, GPU
- Python, C, C++, C#, Ruby, Java (future)

Inferencing with ONNX Runtime

```
import onnxruntime

session =
onnxruntime.InferenceSession("mymodel.onnx")

results = session.run([], {"input": input_data})
```

C#

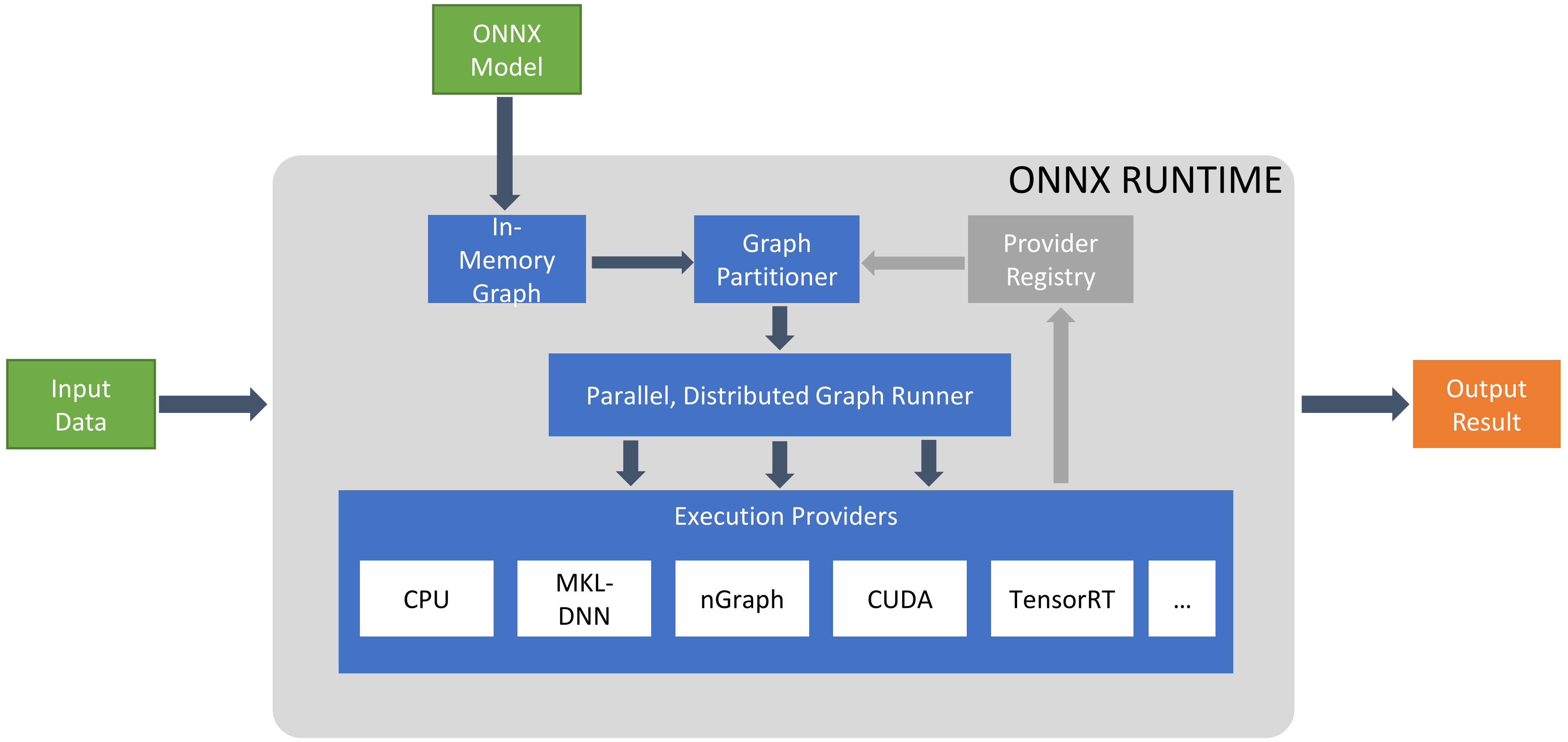
```
using Microsoft.ML.OnnxRuntime;

var session = new InferenceSession("model.onnx");

var results = session.Run(input);
```



Graph optimizations for performance

- Constant folding
- Node eliminations
- Simple and complex node fusions
- Layout optimizations (e.g. NCHWc vs NCHW)
- Extendible and pluggable to add new optimizations

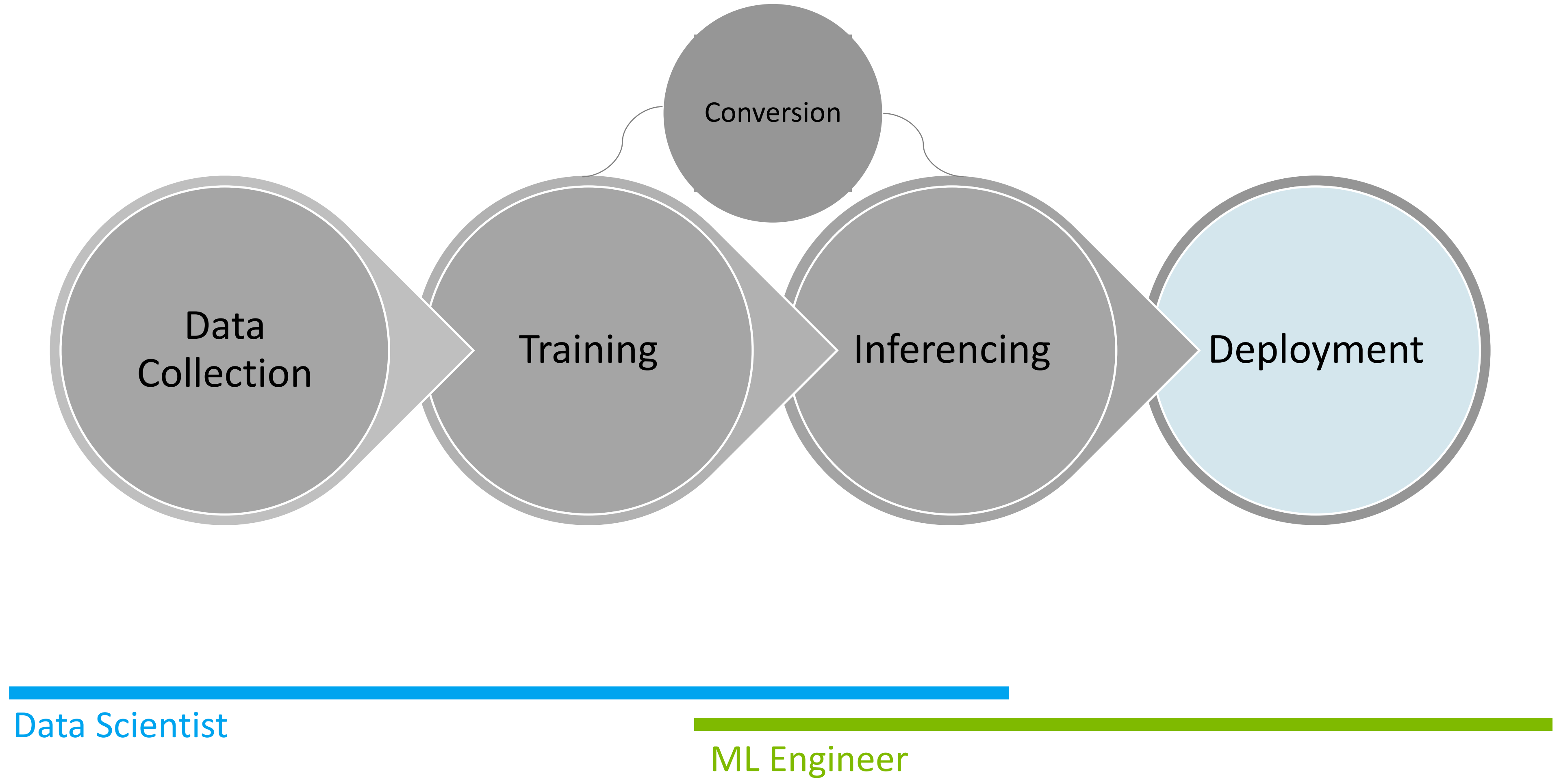
Leverages and abstracts hardware accelerators



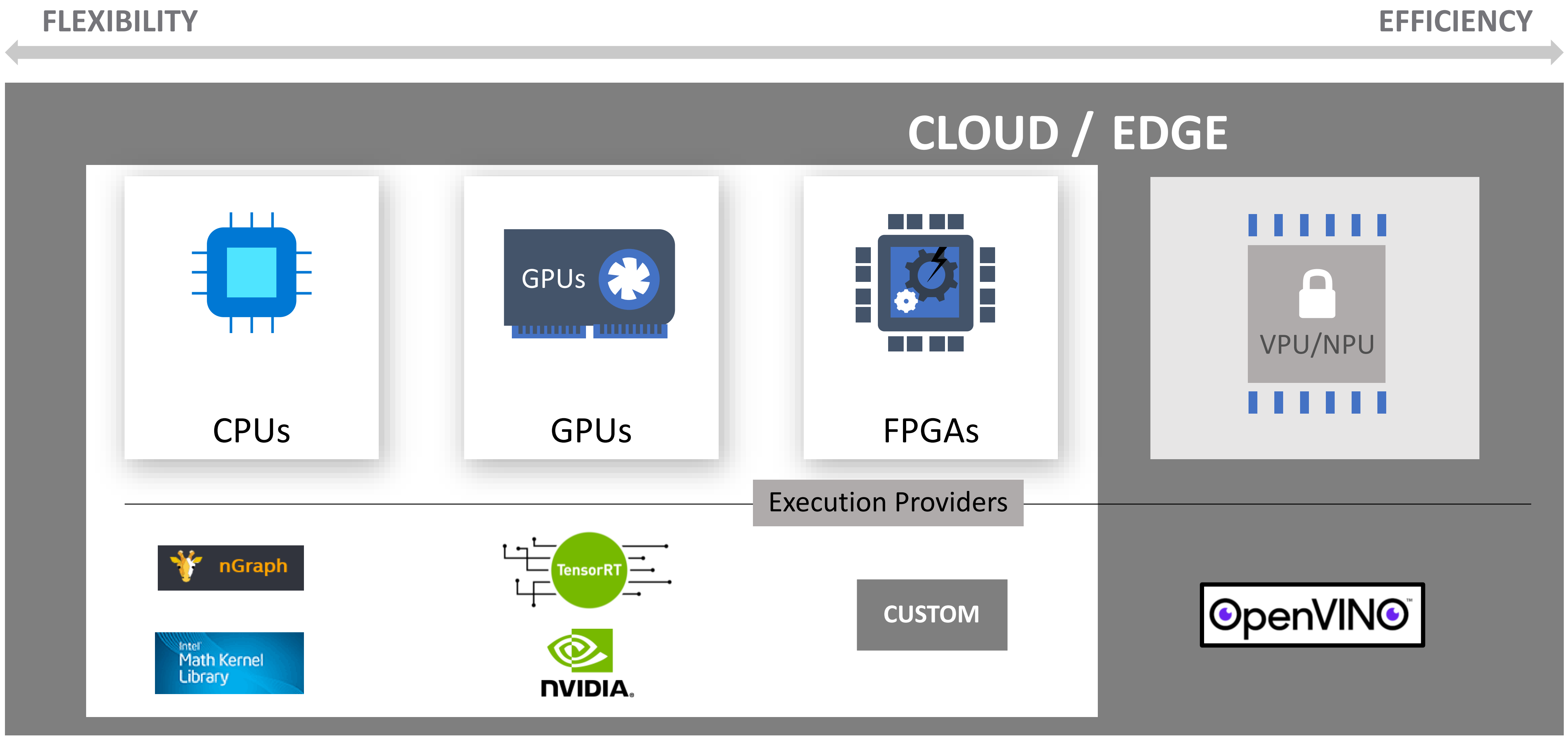
Accelerators for a range of hardware

Base CPU Microsoft Linear Algebra Subprograms	NVIDIA CUDA	NVIDIA TensorRT
Intel nGraph	Intel MKL-DNN	Intel OpenVINO
NUPHAR TVM/LLVM-based model compiler	NN API for Android (future)	DirectML

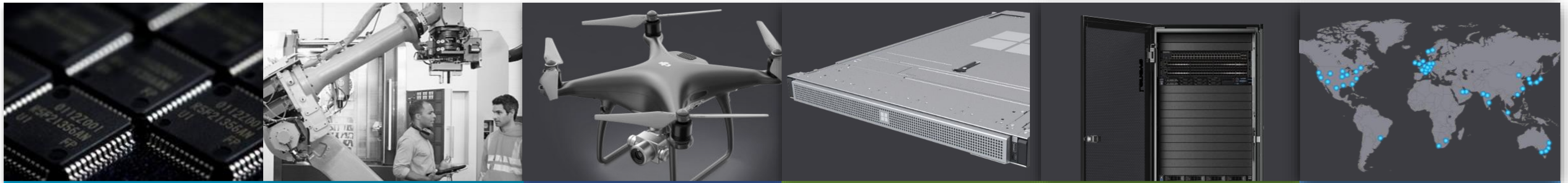
ML Models: Research to Production



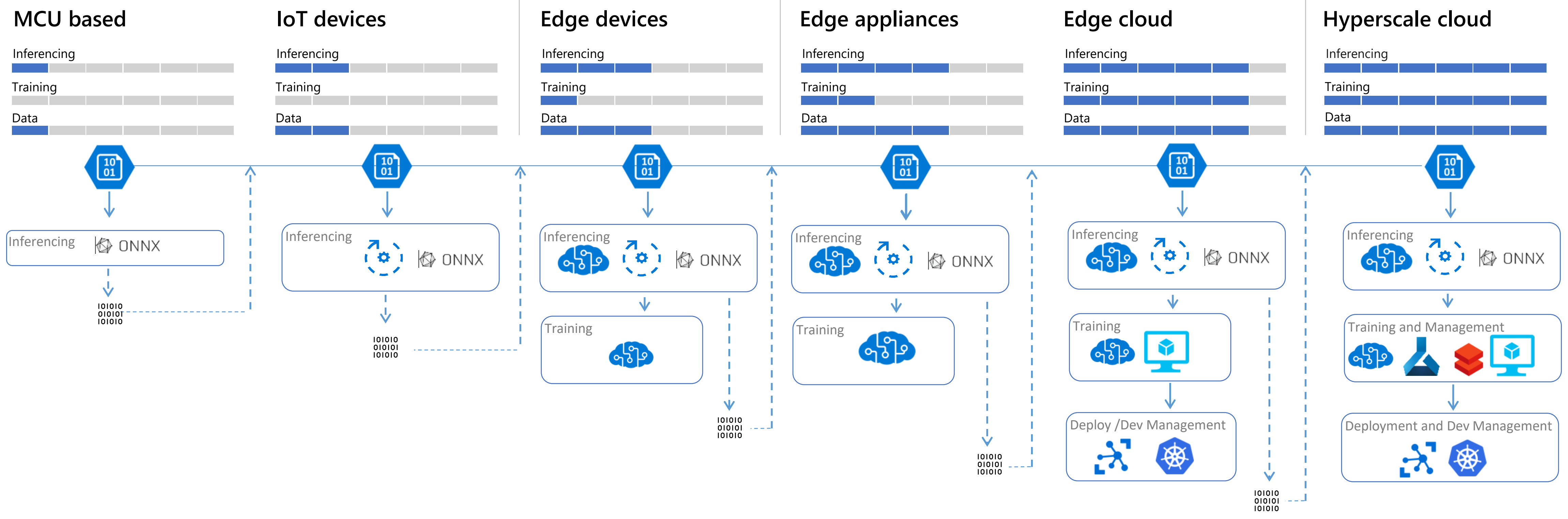
Variety of Deployment Options



Deployment targets with varying compute power



Sensors + control | Sensors to interactive | Integrated platform | Global scale processing



ACTIVITY B

Train a PyTorch model and deploy for production usage

References

- ONNX: https://github.com/onnx/onnx
- ONNX Converters: https://github.com/onnx/onnxmltools/tree/master/onnxmltools
- ONNX Tutorials: https://github.com/onnx/tutorials
- ONNX Runtime: https://github.com/microsoft/onnxruntime
- ONNX Runtime Tutorials: https://github.com/microsoft/onnxruntime#examples-and-tutorials
- Performance Tuning with ONNX Runtime: https://github.com/microsoft/onnxruntime/blob/master/docs/ONNX_Runtime_Perf_Tuning.md
- Training, Inferencing, and deployment in AzureML with ONNX models: https://aka.ms/onnxnotebooks
- AzureML resources: https://azure.microsoft.com/en-us/services/machine-learning/
- Deploying to Edge and IoT devices: Deploying to Intel OpenVINO based devices, Deploying to NVIDIA Jetson Nano (ARM64)
- Windows ML: https://docs.microsoft.com/en-us/windows/ai/windows-ml/

FAITH XU| faxu@microsoft.com

PRABHAT ROY| Prabhat.Roy@microsoft.com