aka.ms/odsc-onnx

Please get started with the pre-requisite

OPEN DATA SCIENCE CONFERENCE

Virtual | Apr. 14 - Apr. 17 2020

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:** Convert the huggingface/transformers BERT model (trained with PyTorch) to ONNX format for accelerated inferencing

  Optional extension: Deploy to a hosted web service using Azure Machine Learning Services

Trends and Growth Areas

Research -> Industry

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

Connectivity, compute, and resources

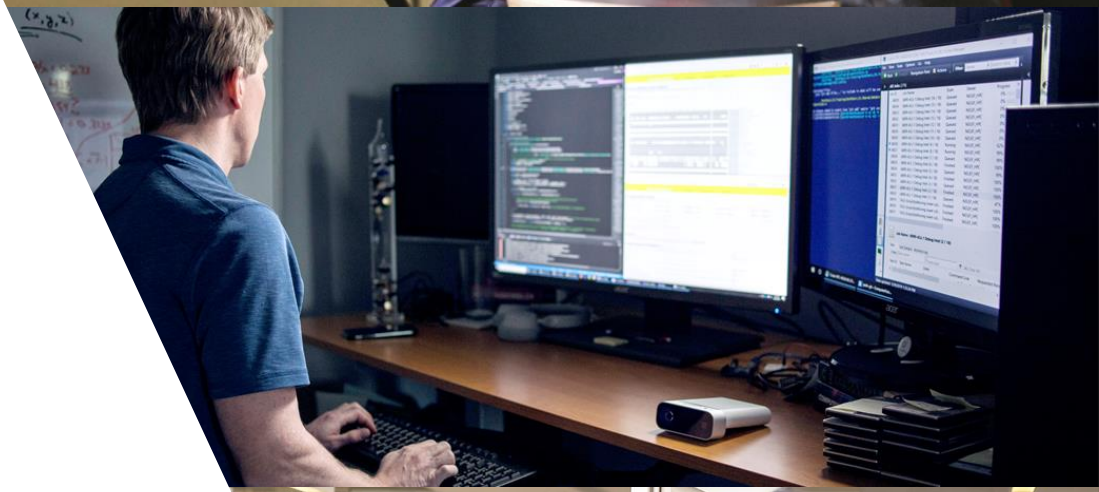
- Infinite storage and compute in the cloud
- CPU, GPUs for training
- LOTS of data

Application spans across all industries

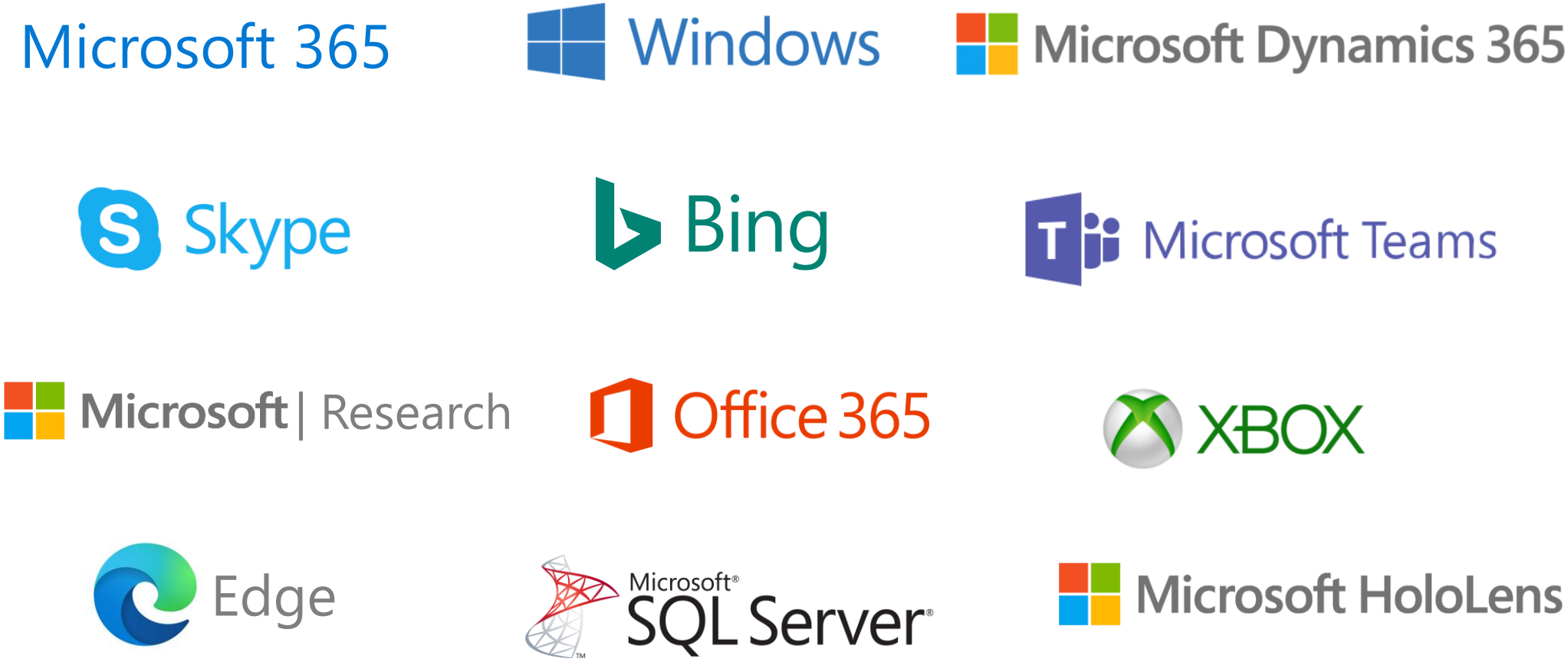
- Healthcare, finance, farming, manufacturing, consumer products, and more

Investments in AI education and jobs

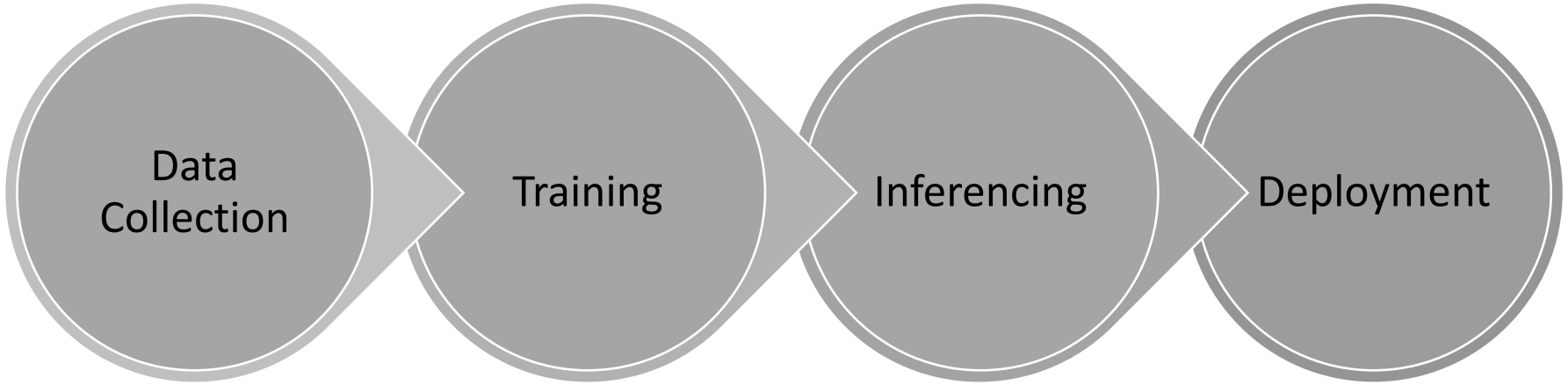
- Universities
- ML Engineer



AI everywhere



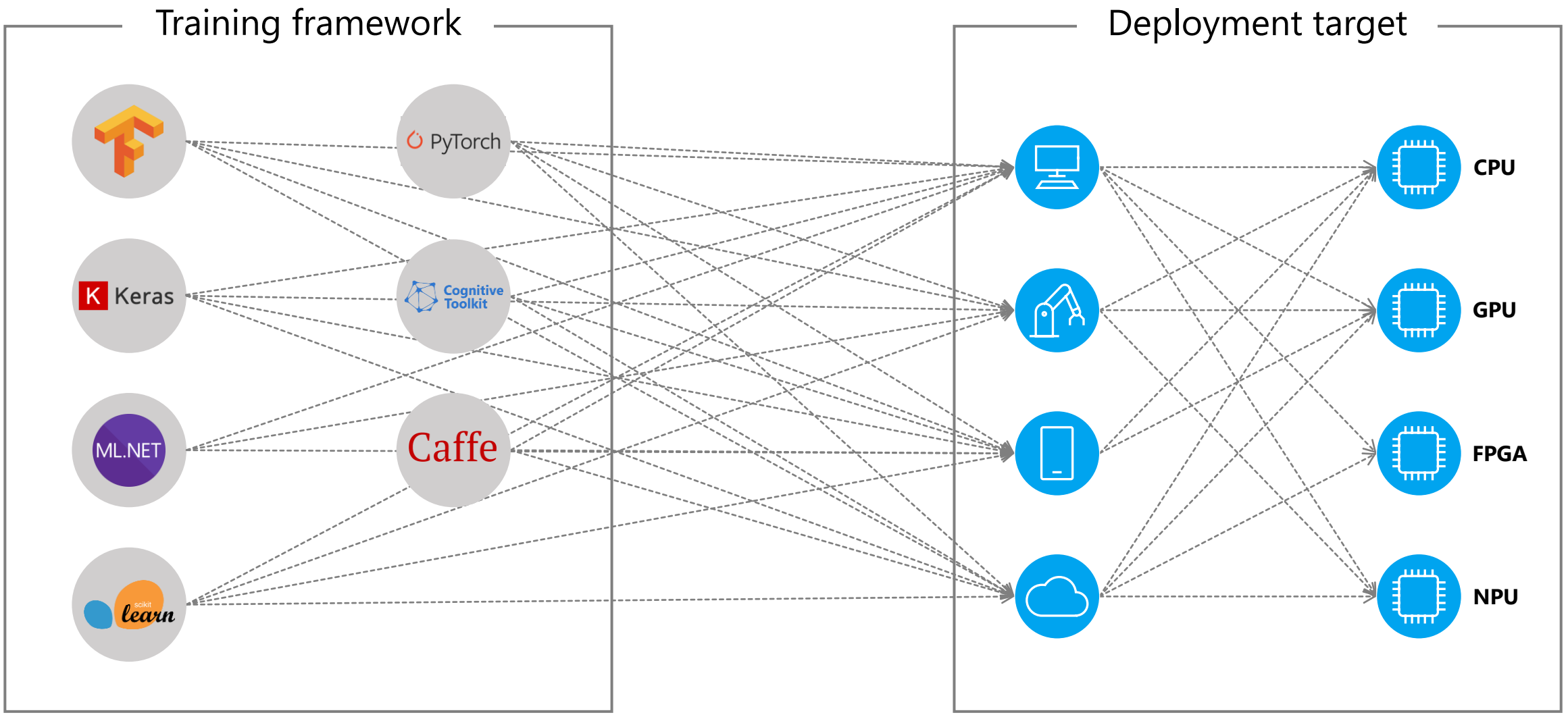
ML Models: Research to Production



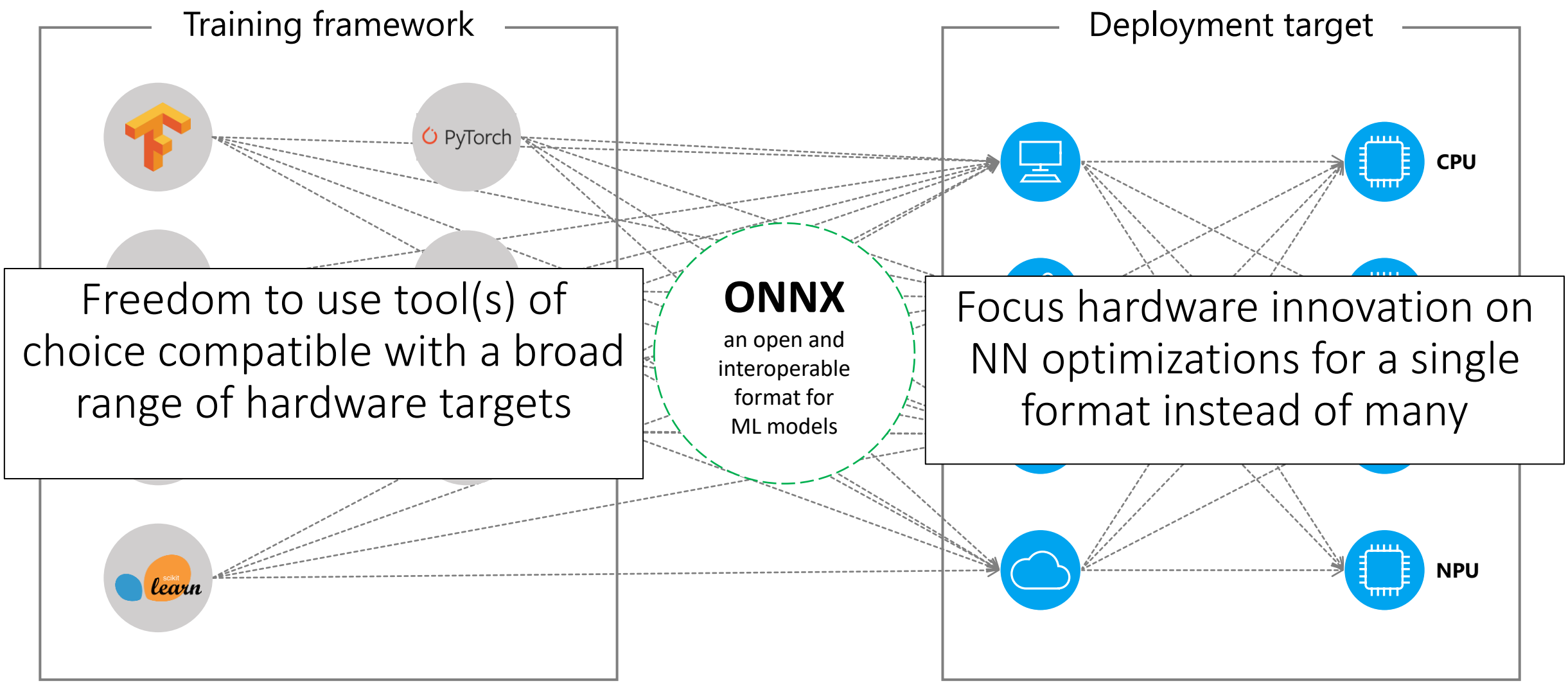
Data Scientist

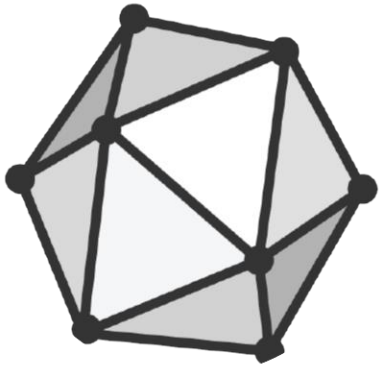
ML Engineer

Training frameworks **x** Deployment targets



ONNX: an open and interoperable format for ML models





ONNX

github.com/onnx

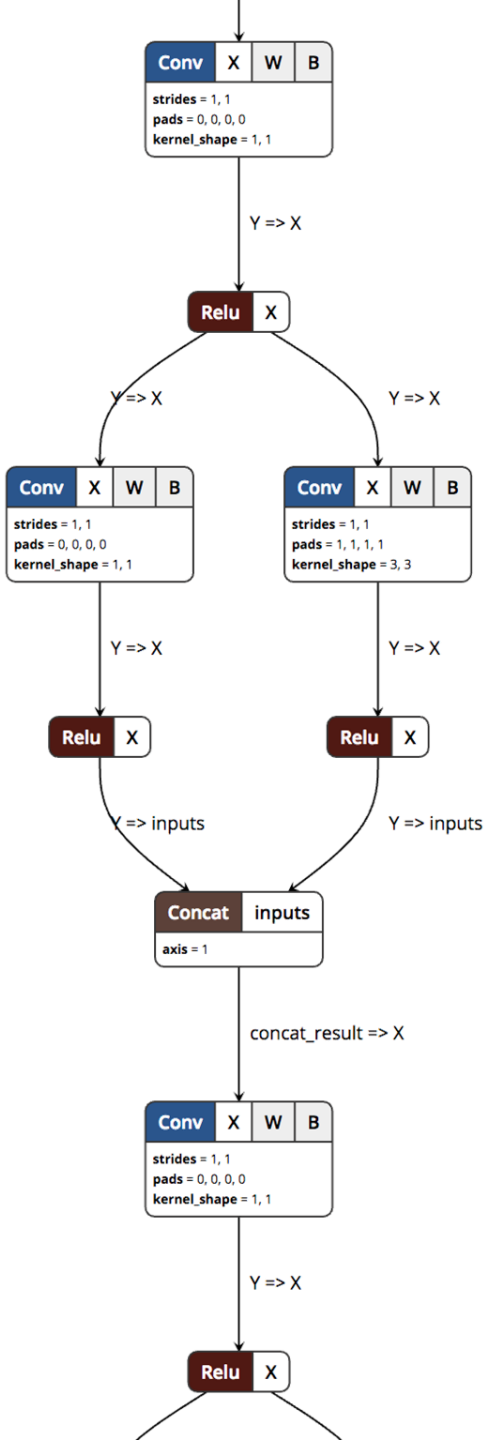
onnx.ai

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

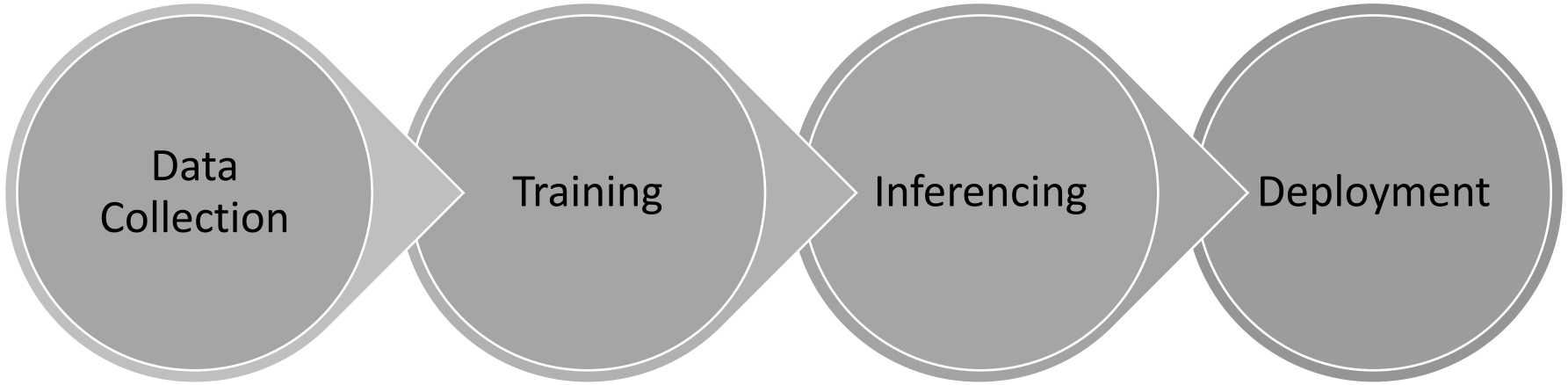

LF AI
Alibaba Group 阿里巴巴集团
AMD
arm
aws
Baidu 百度
BITMAIN
CEVA
Facebook Open Source
GRAPHCORE
habana
Hewlett Packard Enterprise
HUAWEI
IBM
Idein Inc
intel AI
MathWorks
MAXAR
MEDIATEK
Microsoft
Neural Network Libraries
NVIDIA
NXP
Oath: A Verizon company
OctoML
Preferred Networks
Qualcomm
sas
skymizer
SYNOPSYS
Tencent
unity

ONNX

- **Standard** format for ML models consisting of:
  - common Intermediate Representation
  - full operator spec
- Model = graph composed of computational nodes
- Supports both DNN and traditional ML



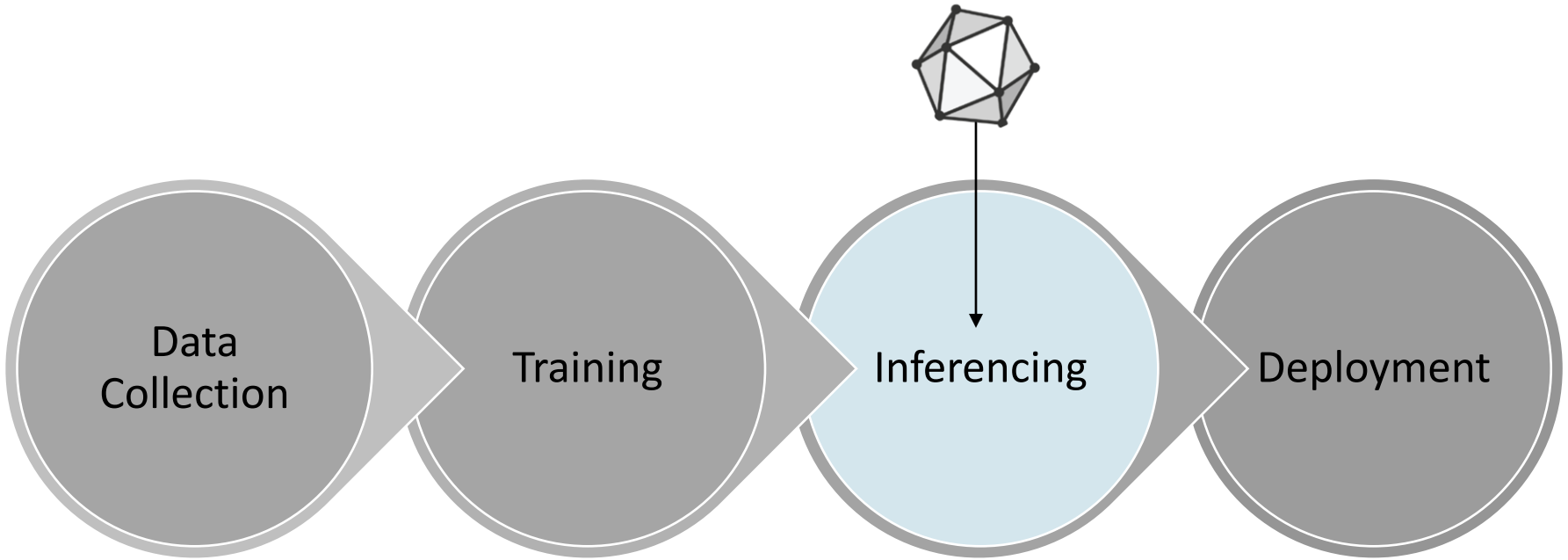
Where does ONNX fit in?



Data Scientist

ML Engineer

Focus on inferencing

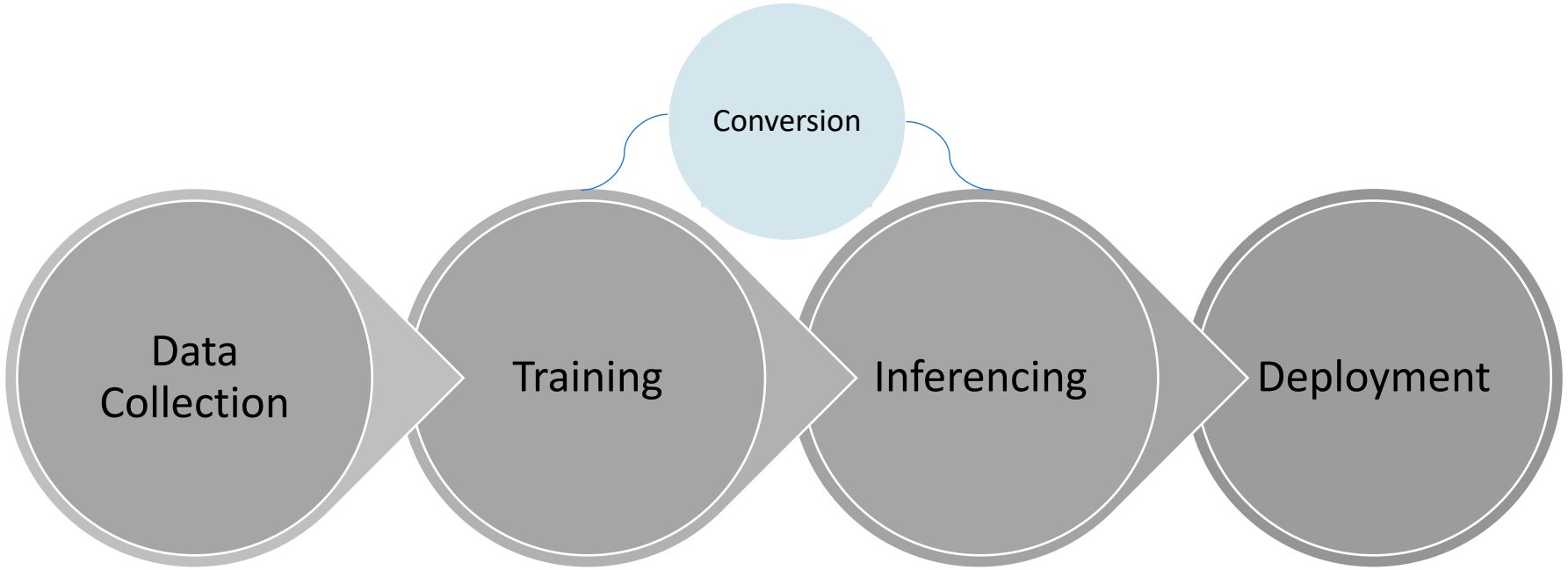


Data Scientist

ML Engineer

Framework Compatibility

How do I get an ONNX model?



Data Scientist

ML Engineer

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

Open Source converters for popular ML 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
H2O: onnx/onnxmltools
ML.NET (native export)
SparkML (experimental): 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')
```

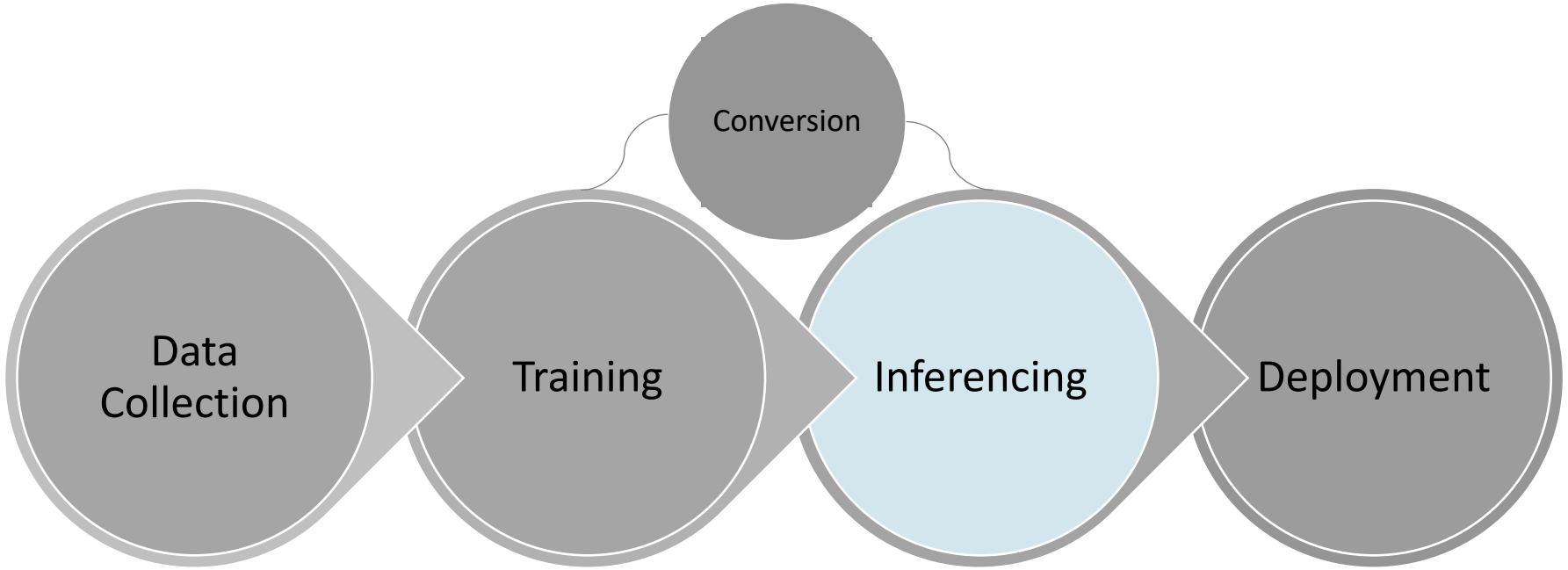
```
import torch
import torch.onnx

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")
```

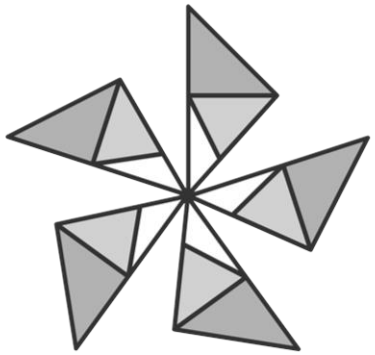
Inferencing ONNX models

ML Models: Research to Production



Data Scientist

ML Engineer



ONNX Runtime

aka.ms/onnxruntime

github.com/microsoft/onnxruntime

ONNX Runtime: open source high performance Inference Engine

- Performance focused design
- Full ONNX operator support
- Flexibility for custom operators not in the spec
- Backwards compatible to minimize versioning issues with software or model updates

Cross platform, multi language API

OS	Windows	Linux	Mac		
Language	Python (3.5-3.7)	C++	C#	C	Java
Architecture	X64	X86	ARM64	ARM32	
Hardware Acceleration	Default CPU	CUDA	TensorRT	DirectML	MKL-DNN
	MKL-ML	nGraph	NUPHAR	OpenVINO	
Installation Instructions	pip install onnxruntime				

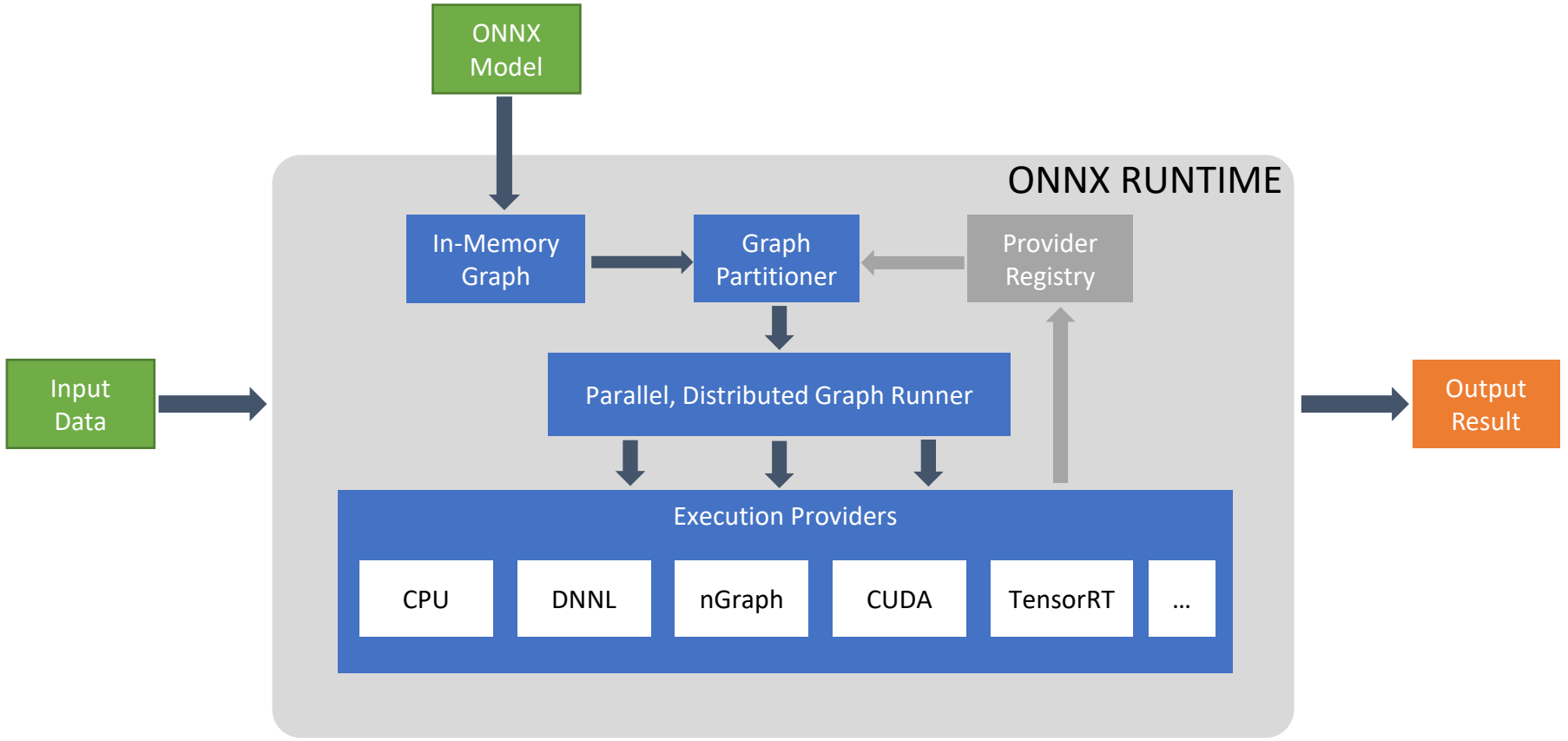
Inferencing with ONNX Runtime

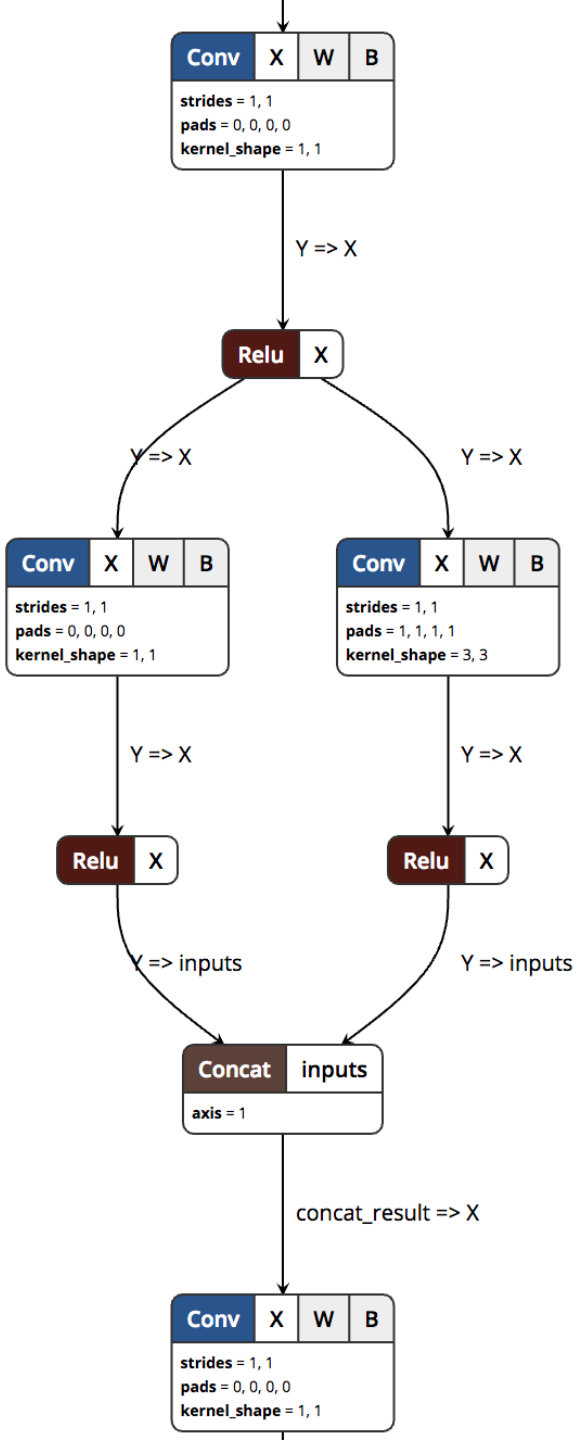
```python
import onnxruntime

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);
```

Leverages and abstracts hardware accelerators

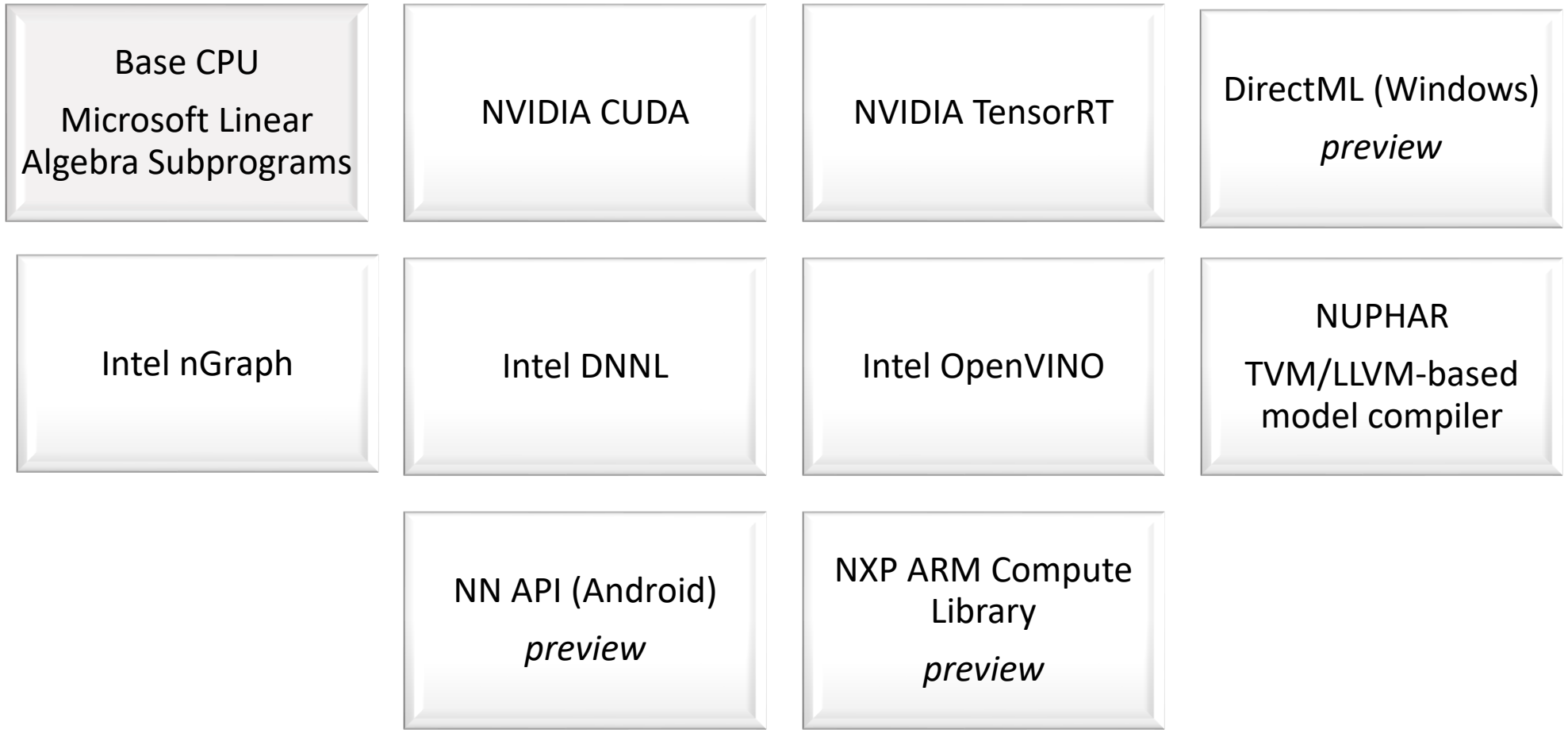




Graph Optimizations

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

Accelerators for a range of hardware

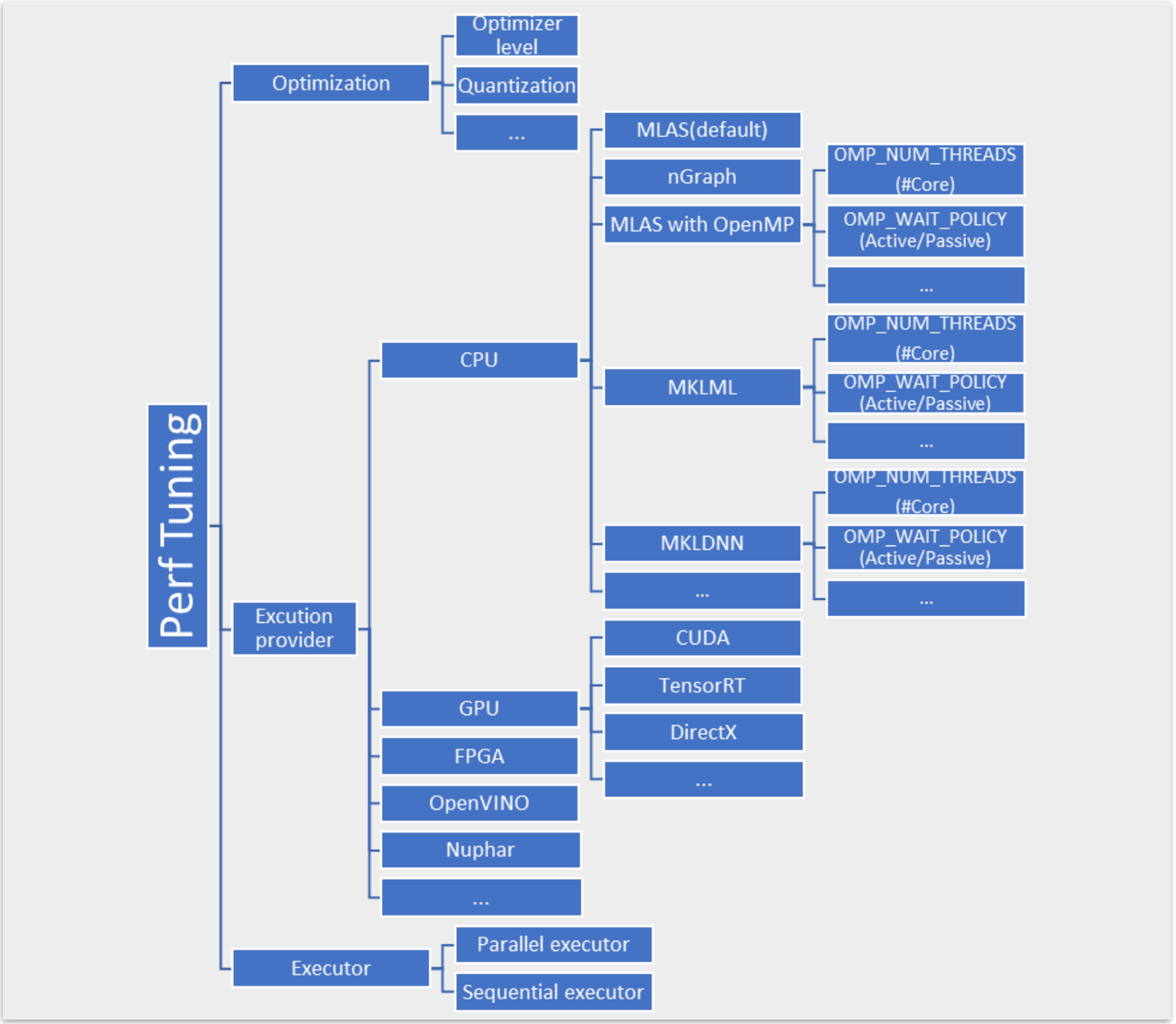


Perf Tuning in ONNX Runtime

ONNX Go Live (OLive)

https://github.com/microsoft/OLive

- Automates the process of ONNX model shipping by integrating model conversion, correctness test, and performance tuning into a single pipeline
- Outputs a production ready ONNX model with ONNX Runtime configurations (execution provider + optimization options)

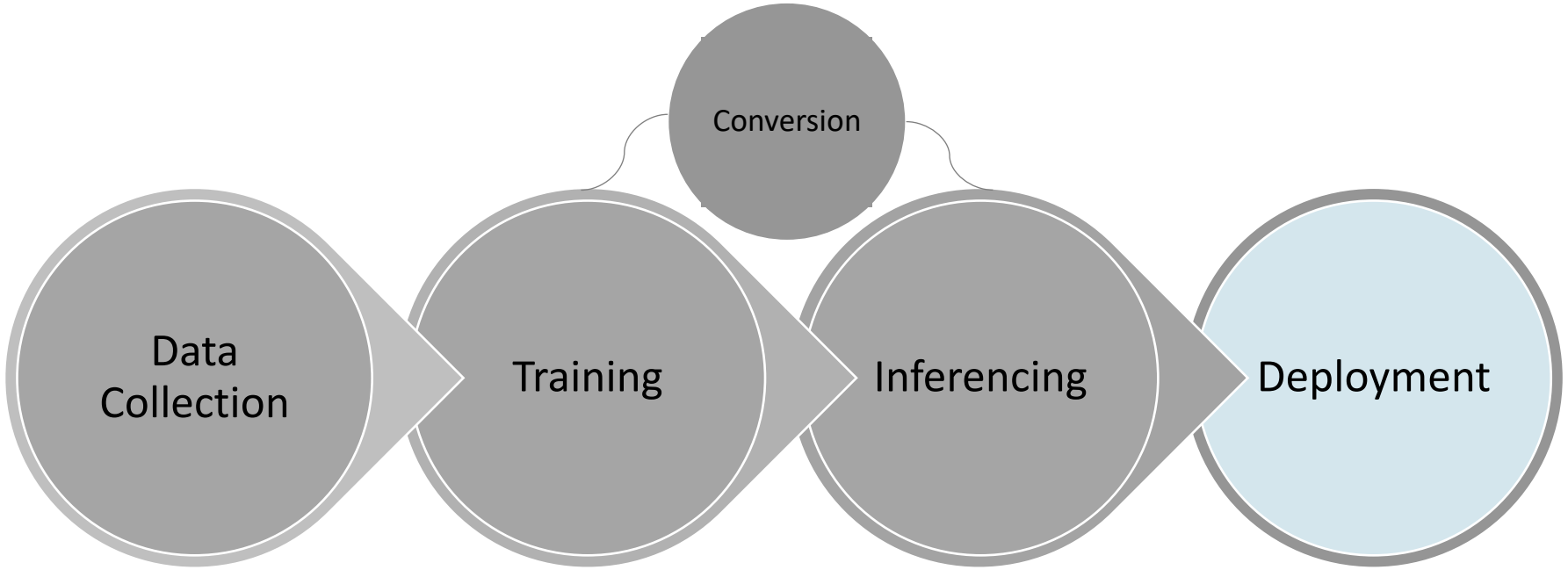


Focus areas and further investments

- ONNX spec alignment
- Performance optimizations
- Hardware ecosystem and mobile/edge/IoT devices
- Production models
- Training acceleration

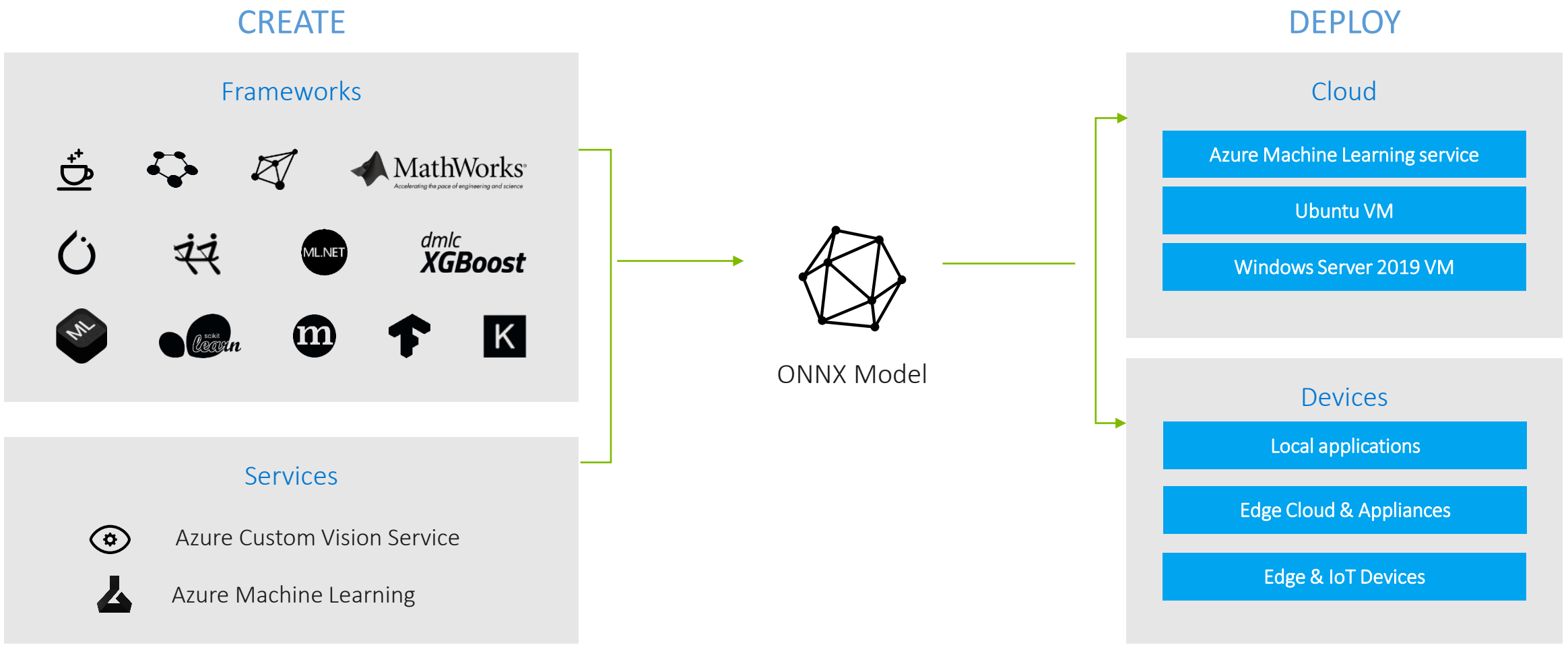
Deploying models with ONNX Runtime

ML Models: Research to Production

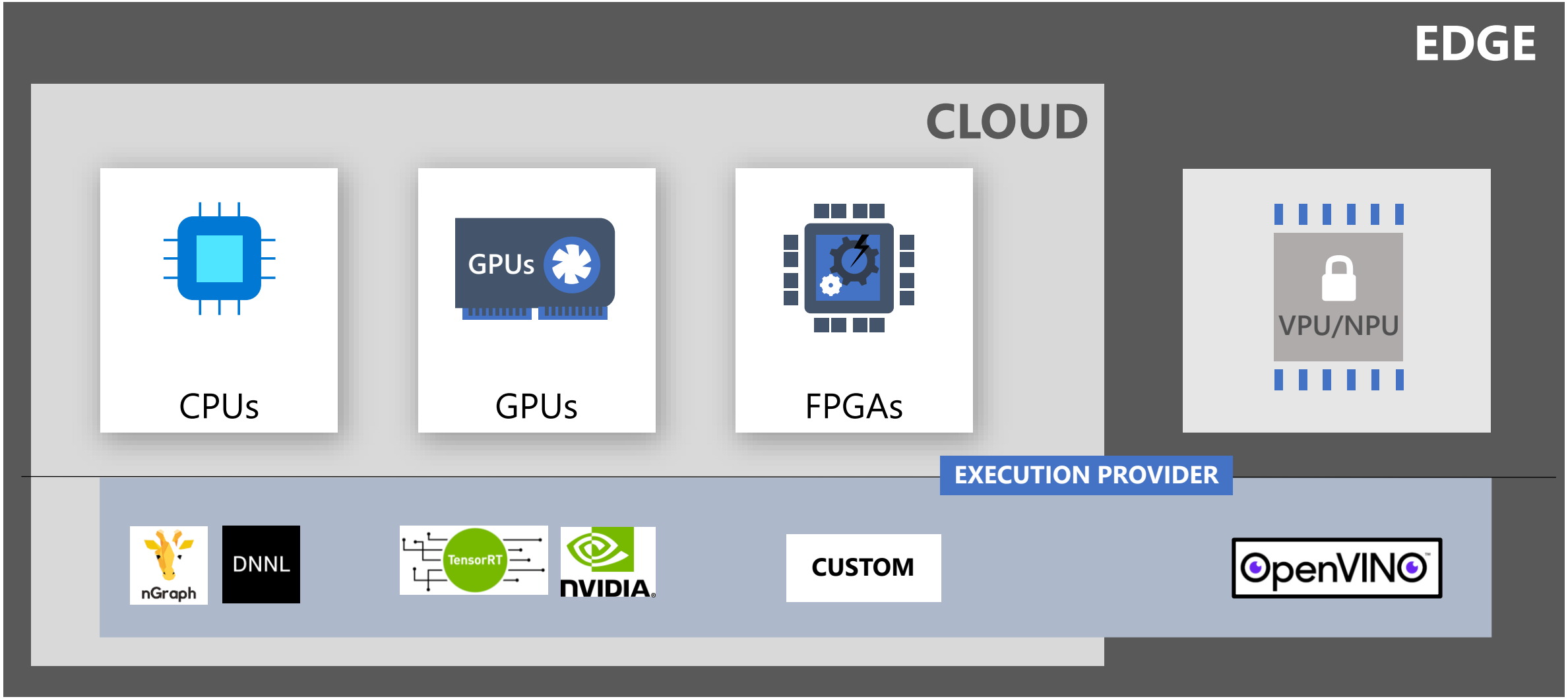


Data Scientist

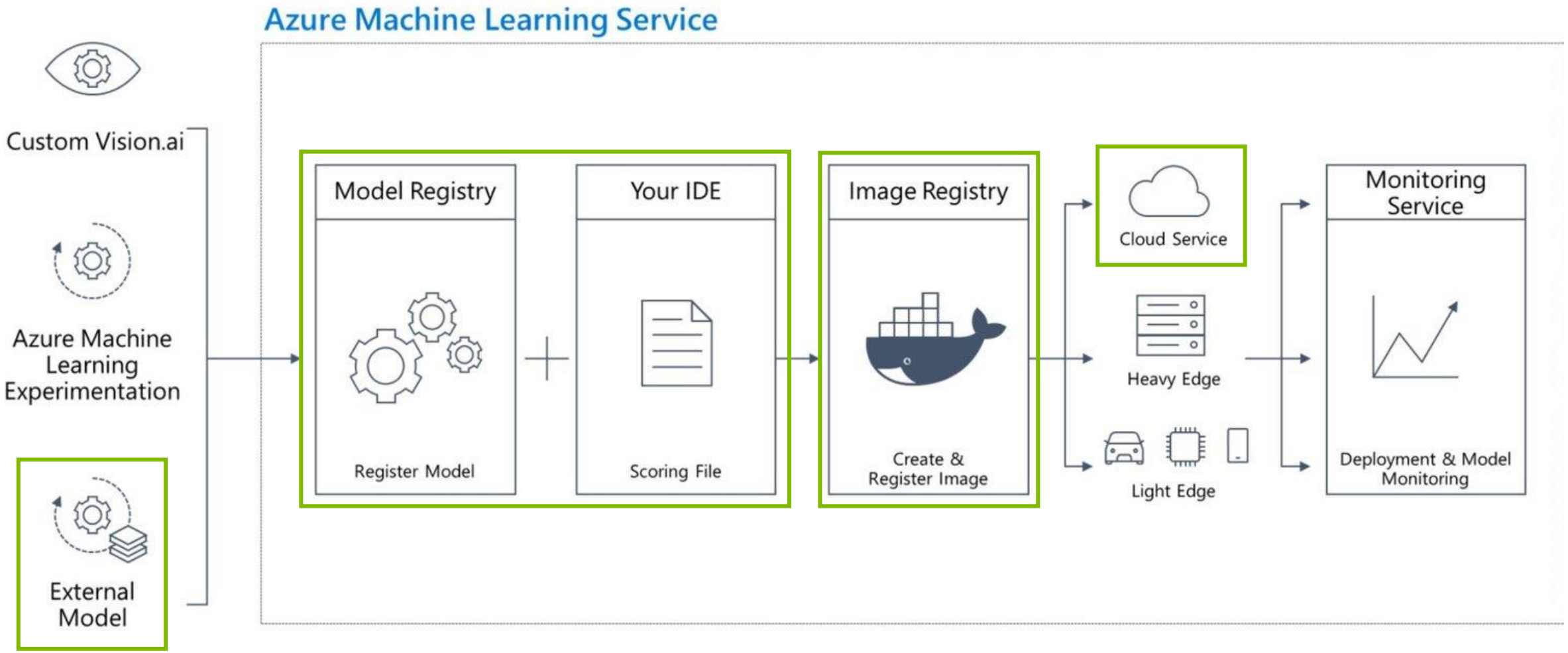
ML Engineer


CREATE
Frameworks
MathWorks
Accelerating the pace of engineering and science
ML.NET
dmlc
XGBoost
scikit
learn
K
Services
Azure Custom Vision Service
Azure Machine Learning
ONNX Model
DEPLOY
Cloud
Azure Machine Learning service
Ubuntu VM
Windows Server 2019 VM
Devices
Local applications
Edge Cloud & Appliances
Edge & IoT Devices

Variety of Deployment Options



Deployment through Azure ML



ACTIVITY

Convert the huggingface/transformers BERT model (trained with PyTorch) to ONNX format for accelerated inferencing

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