End-to-end deep learning deployment with ONNX

Nick Pentreath
Principal Engineer

@MLnick

About

– *@MLnick* on Twitter & Github

– Principal Engineer, IBM CODAIT (Center for Open-Source Data & AI Technologies)

– Machine Learning & AI

– Apache Spark committer & PMC

– Author of *Machine Learning with Spark*

– Various conferences & meetups

Center for Open Source Data & AI Technologies

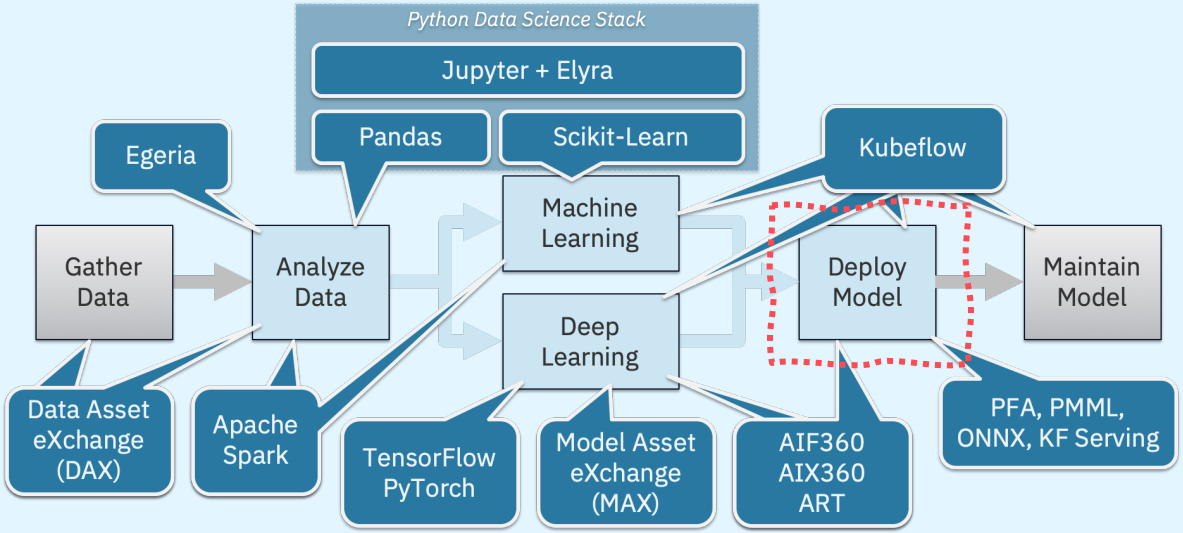
CODAIT

Open Source @ IBM

codait.org

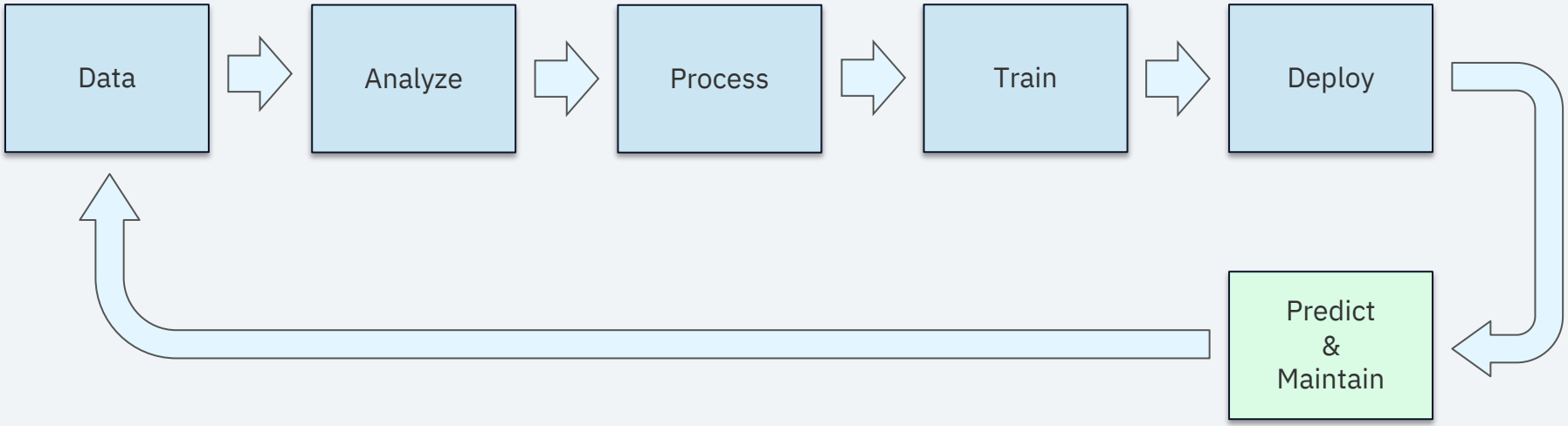
- CODAIT aims to make AI solutions dramatically easier to create, deploy, and manage in the enterprise.
- We contribute to and advocate for the open-source technologies that are foundational to IBM's AI offerings.
- 30+ open-source developers!

Improving the Enterprise AI Lifecycle in Open Source

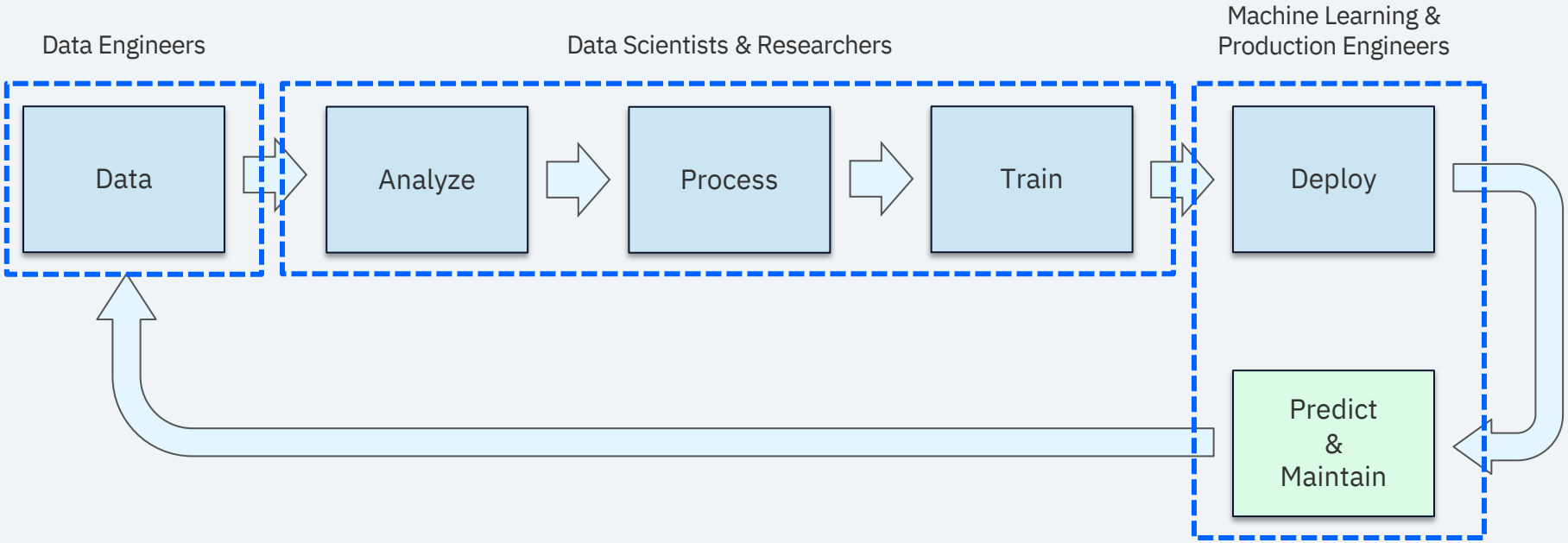


The Machine Learning Workflow

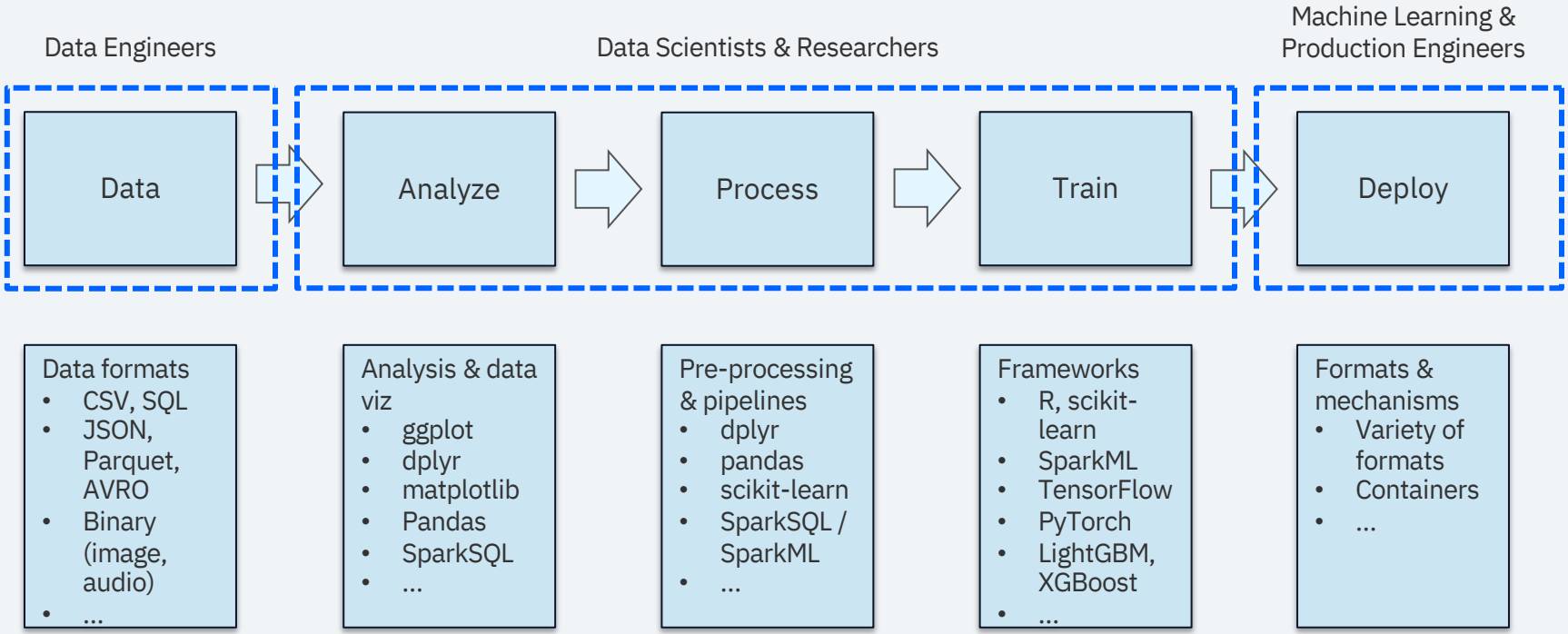
Machine Learning Workflow



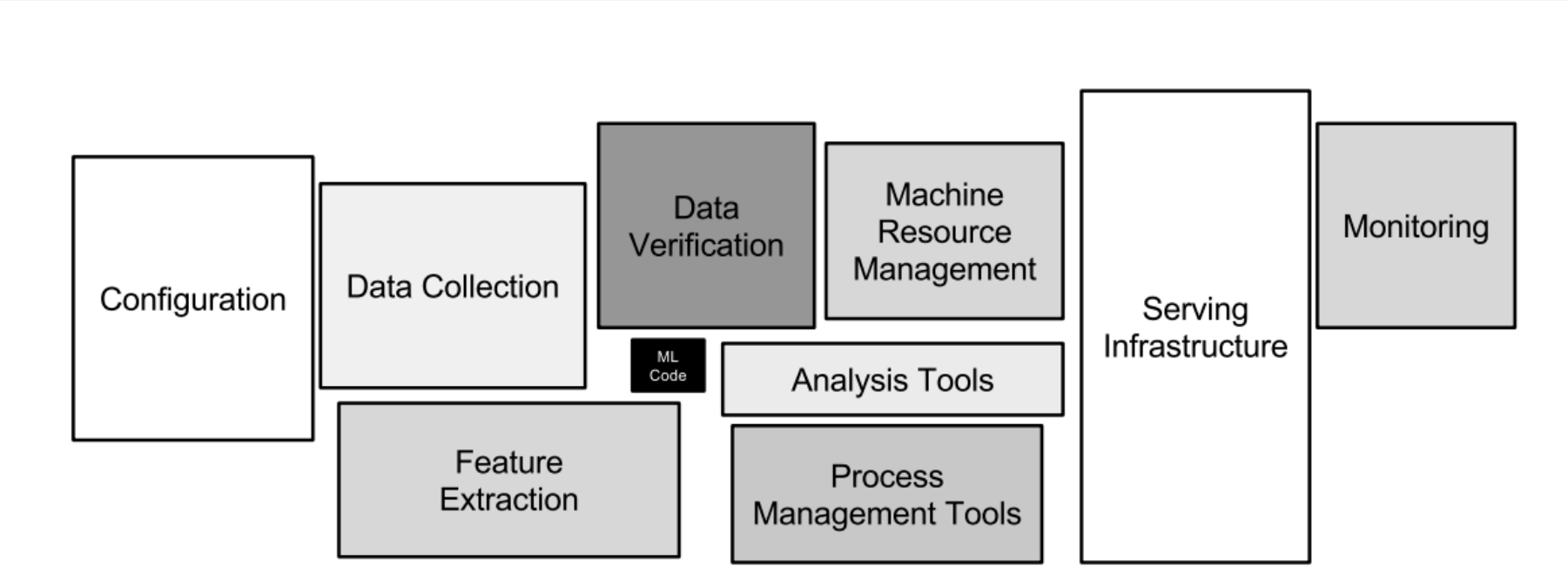
Workflow spans teams …



… and tools …


Data Engineers
Data Scientists & Researchers
Machine Learning & Production Engineers
Data
Analyze
Process
Train
Deploy
Data formats
• CSV, SQL
• JSON, Parquet, AVRO
• Binary (image, audio)
• …
Analysis & data viz
• ggplot
• dplyr
• matplotlib
• Pandas
• SparkSQL
• …
Pre-processing & pipelines
• dplyr
• pandas
• scikit-learn
• SparkSQL / SparkML
• …
Frameworks
• R, scikit-learn
• SparkML
• TensorFlow
• PyTorch
• LightGBM, XGBoost
• …
Formats & mechanisms
• Variety of formats
• Containers
• …

… and is a small (but critical!) piece of the puzzle



*Source: Hidden Technical Debt in Machine Learning Systems

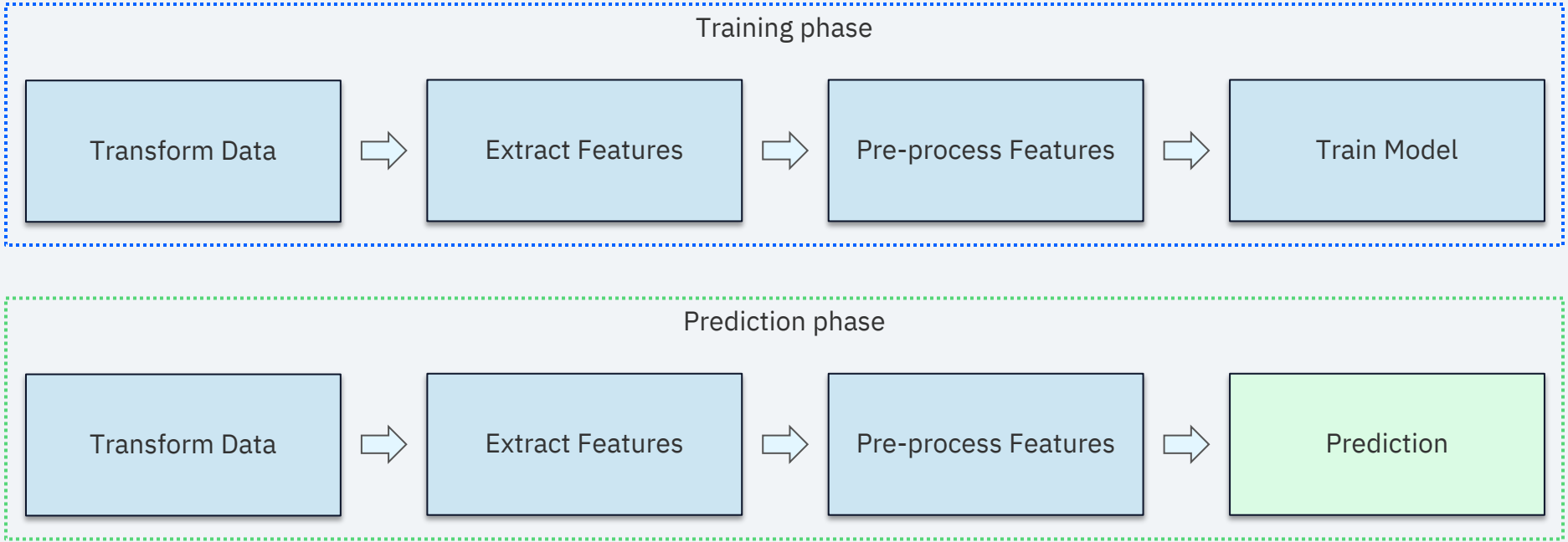
Machine Learning Deployment

What, Where, How?

- What are you deploying?
  - What is a “model”
- Where are you deploying?
  - Target environment (cloud, browser, edge)
  - Batch, streaming, real-time?
- How are you deploying?
  - Devops / serving frameworks

We will talk mostly about the what

What is a “model”?



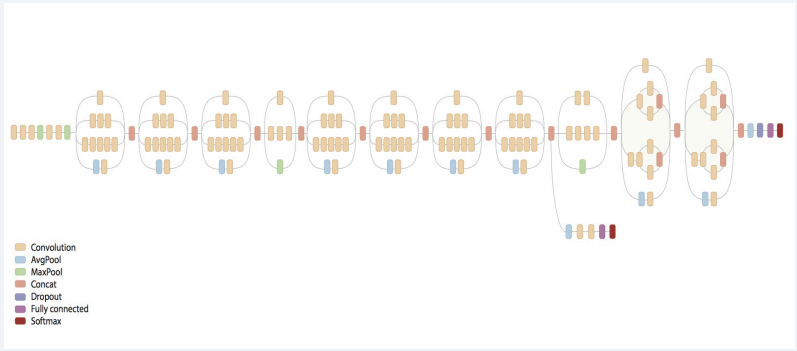
Deep Learning doesn't need feature engineering or data processing …

right?

Deep learning pipeline?

Input image

Inference



Prediction

beagle: 0.82

Source: https://ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html

Deep learning pipeline!

Input image

Image pre-processing

- Decode image
- Resize
- Normalization
- Convert types / format

PIL, OpenCV, tf.image, ...

Inference

Post-processing

- [0.2, 0.3, ...]
- Label map
- (label, prob)
- Sort

Custom Python

Prediction

```
beagle: 0.82
basset: 0.09
bluetick: 0.07
...
```


** Logos trademarks of their respective projects*

Image pre-processing

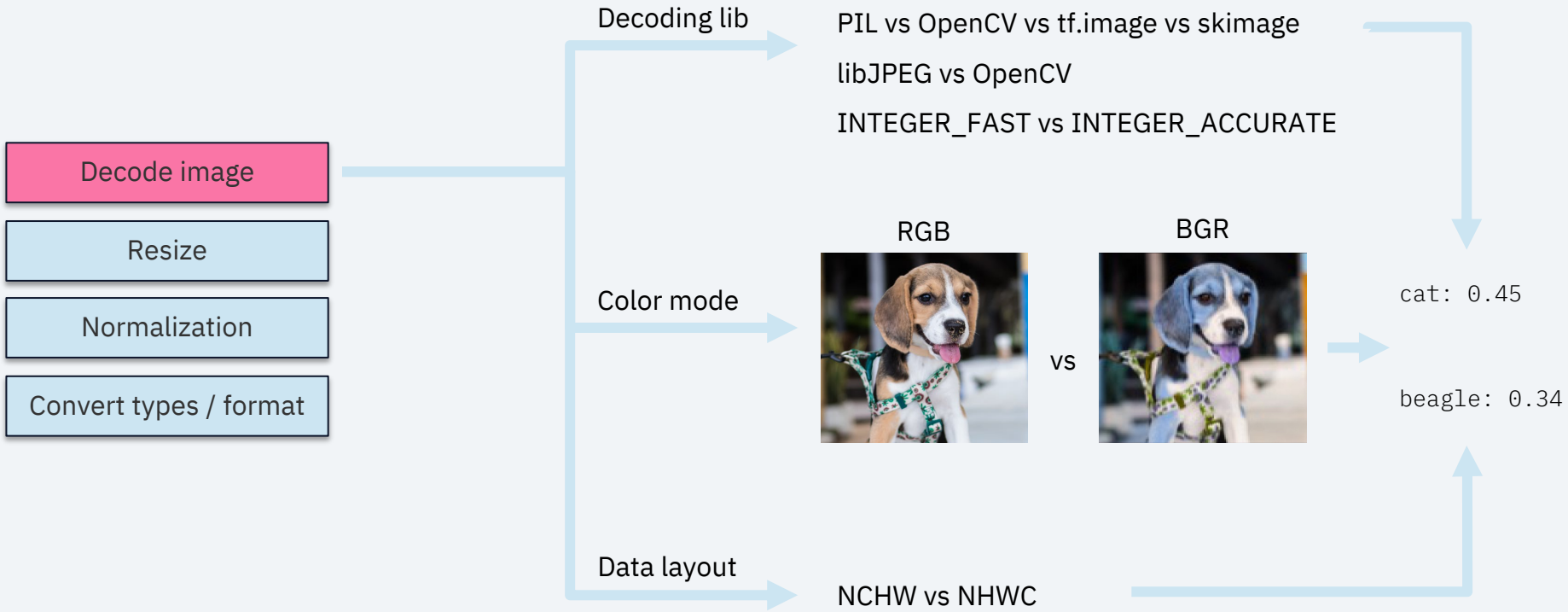
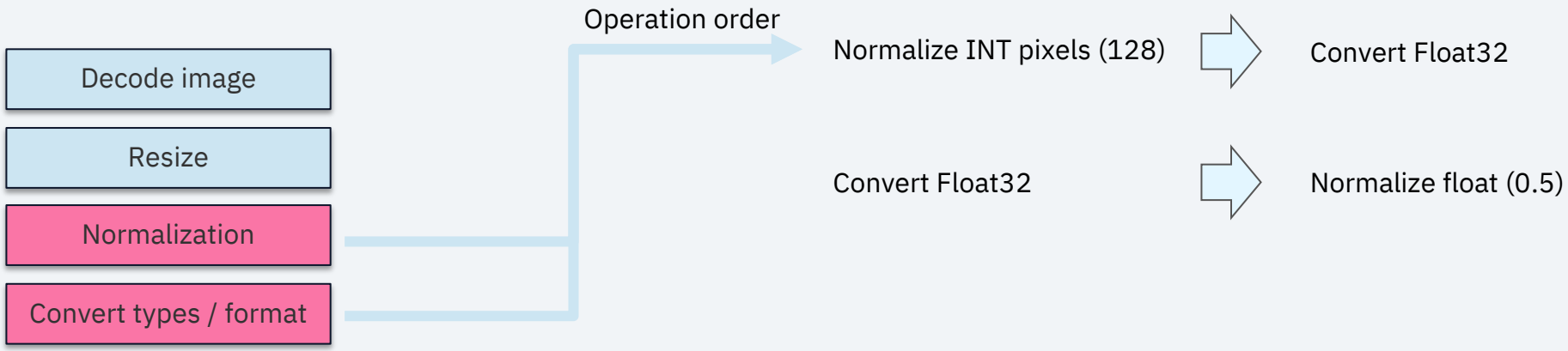
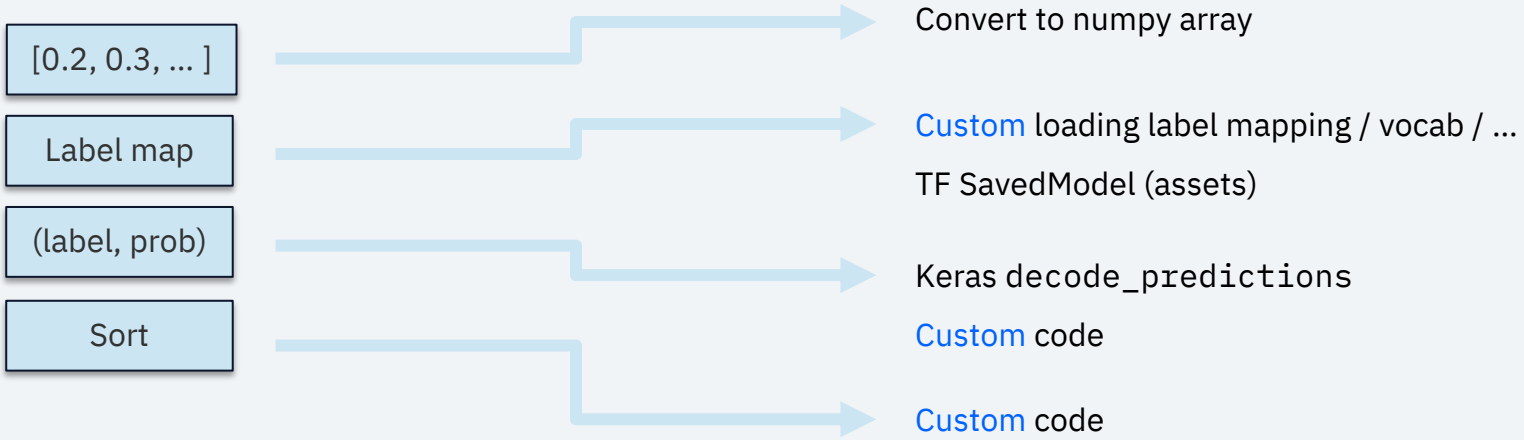


Image pre-processing



Inference post-processing



Pipelines, not Models

- Deploying just the model is not enough

- Entire pipeline must be deployed
  - Data transforms
  - Feature extraction & pre-processing
  - DL / ML model
  - Prediction transformation

- Even ETL is part of the pipeline!

Challenges

– Manage and bridge:
  - Languages
  - Frameworks
  - Dependencies / versions
– Performance can vary across these dimensions
– Friction between teams
  - Data scientists & researchers
  - Production
  - Business

– Formats
  - Each framework does things differently
  - Proprietary formats: lock-in, not portable
– Custom solutions and extensions

** Logos trademarks of their respective projects*

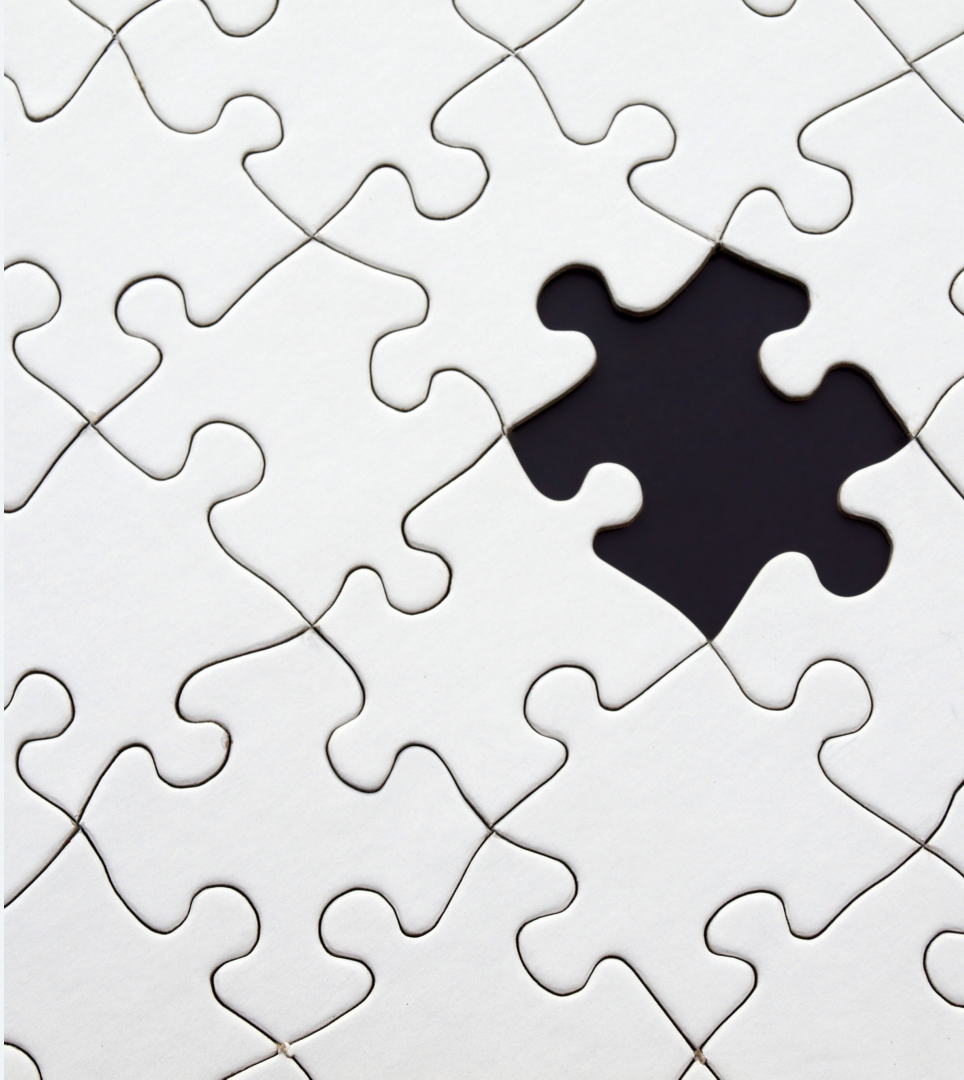
Containers for ML Deployment

Containers are “The Solution” … right?

– Container-based deployment has significant benefits
  - Repeatability
  - Ease of configuration
  - Separation of concerns – focus on what, not how
  - Allow data scientists & researchers to use their language / framework of choice
  - Container frameworks take care of (certain) monitoring, fault tolerance, HA, etc.

– But …
  - What goes in the container is still the most important factor
  - Performance can be highly variable across language, framework, version
  - Requires devops knowledge, CI / deployment pipelines, good practices
  - Does not solve the issue of standardization
    - Formats
    - APIs exposed
  - A serving framework is still required on top

Open Standards for Model Serialization & Deployment



Why a *standard*?

Standard Format

Single stack

Execution

Optimization

Tooling (Viz, analysis, ...)

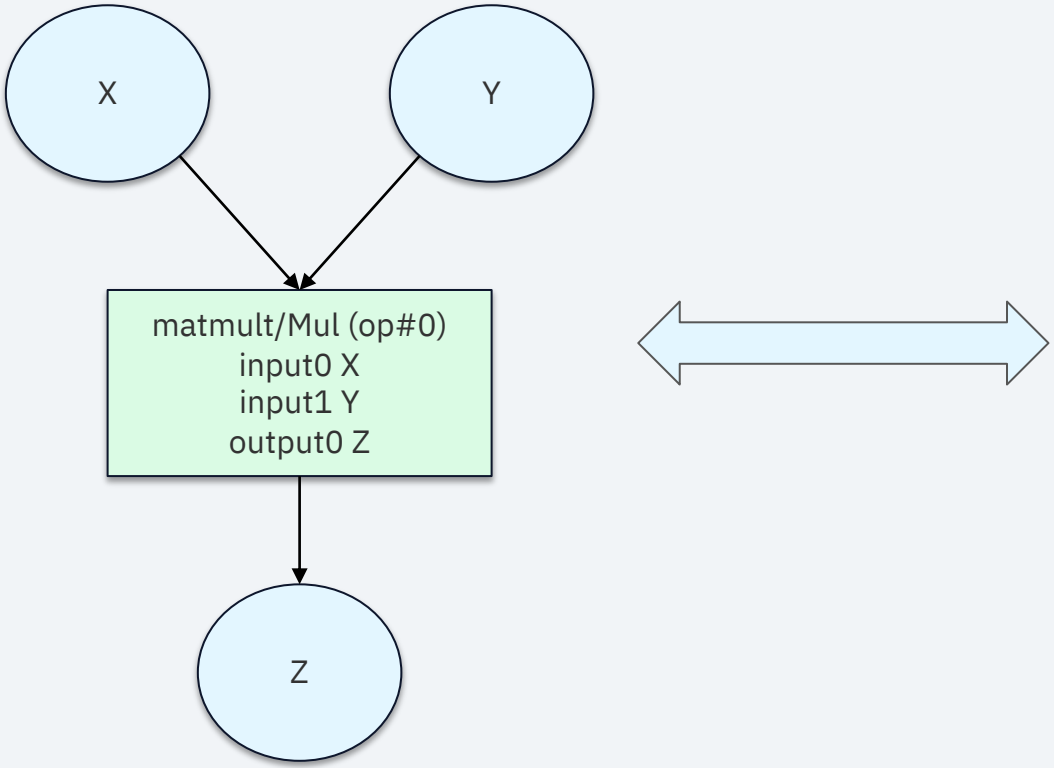
Why an *Open Standard*?

- Open-source vs open standard
- Open source (license) is only one aspect
  - OSS licensing allows free use, modification
  - Inspect the code etc
  - … but may not have any *control*
- Open governance is critical
  - Avoid concentration of control (typically by large companies, vendors)
  - Visibility of development processes, strategic planning, roadmaps
- However there are downsides
  - Standard needs wide adoption and critical mass to succeed
  - A standard can move slowly in terms of new features, fixes and enhancements
  - Design by committee
  - Keeping up with pace of framework development

Open Neural Network Exchange (ONNX)

- Established 2017 by Facebook & Microsoft
- Protobuf for serialization format and type specification
- Describes
  - computation graph (inputs, outputs, operators) - DAG
  - values (weights)
- Serialized graph is self-contained
- Focused on Deep Learning / tensor operations
- Baked into PyTorch from 1.0.0 / Caffe2 as the serialization & interchange format

ONNX Graphs

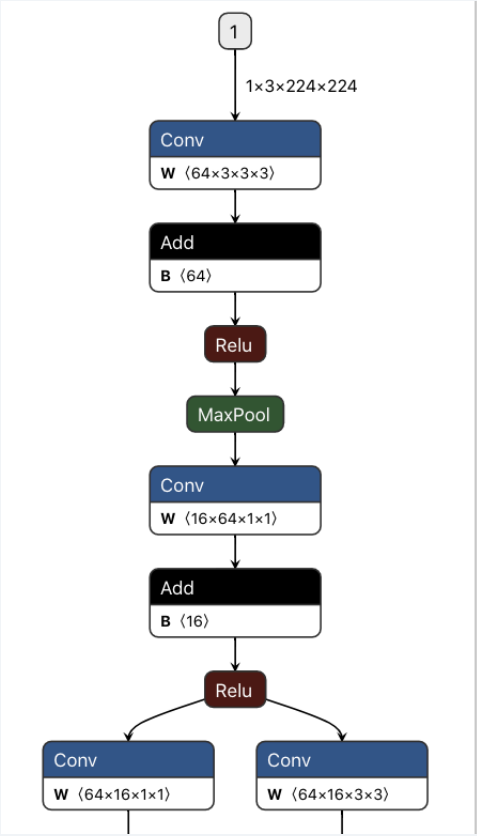


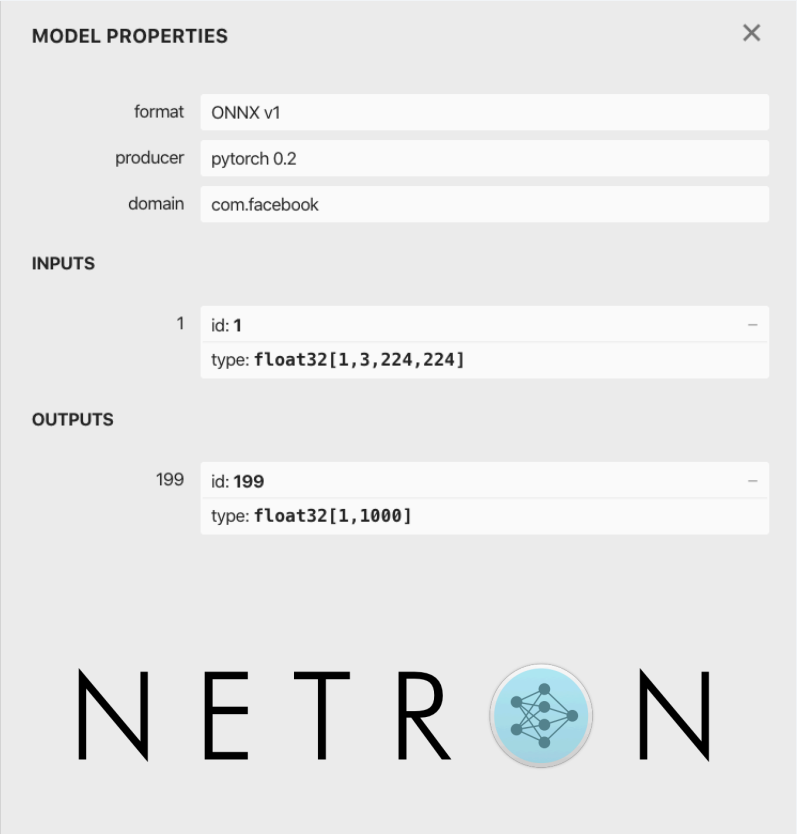

```
graph {
  node {
    input: "X"
    input: "Y"
    output: "Z"
    name: "matmult"
    op_type: "Mul"
  }
  input {
    name: "X"
    type { ... }
  }
  output {
    name: "Z"
    type { ... }
  }
}
```


Source: http://onnx.ai/sklearn-onnx/auto_examples/plot_pipeline.html

ONNX Graphs

SqueezeNet
Computation Graph
Visualization





Link: https://github.com/lutzroeder/Netron#models

Deep Learning Framework Support

– Framework converters
  - PyTorch / Caffe2 (baked in)
  - TensorFlow
  - Keras
  - Apple CoreML
  - MXNet
  - MS Cognitive Toolkit
  - … many more

https://onnx.ai/supported-tools.html

ONNX-ML

– Provides support for (parts of) "traditional" machine learning
  - Additional types: sequences, maps
  - Operators
    - Vectorizers (numeric & string data)
    - One hot encoding, label encoding
    - Scalers (normalization, scaling)
    - Models (linear, SVM, TreeEnsemble)
    - …

https://github.com/onnx/onnx/blob/master/docs/Operators-ml.md

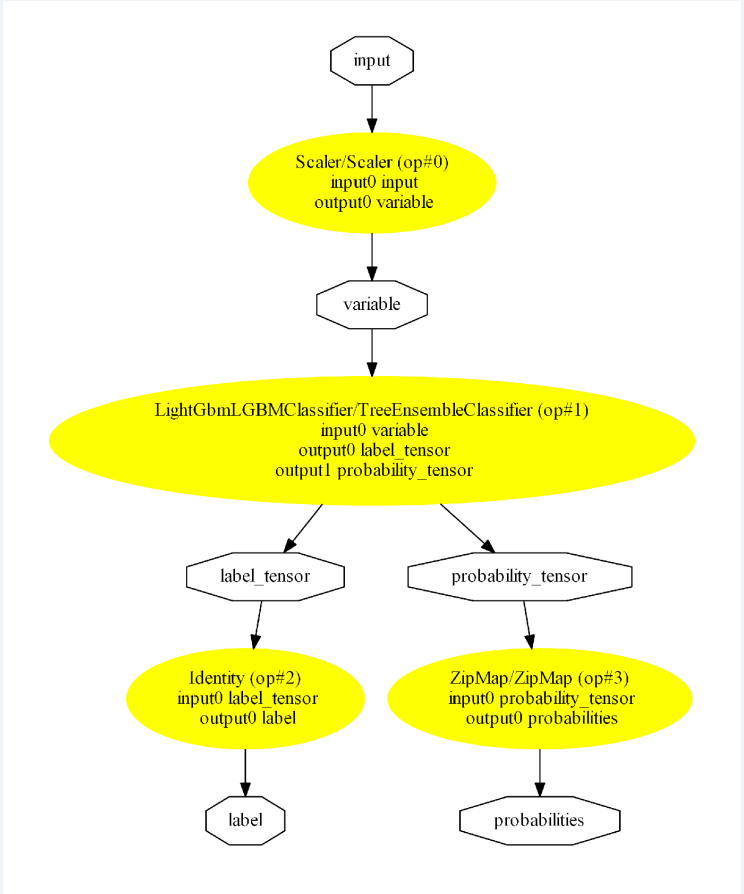
ONNX-ML

– Exporter support
  - Scikit-learn – 60+
  - LightGBM
  - XGBoost
  - Apache Spark ML – 25+
  - Keras – all layers + TF custom layers
  - Libsvm
  - Apple CoreML

https://github.com/onnx/onnxmltools/

http://onnx.ai/sklearn-onnx/index.html

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



ONNX Ecosystem

Converters

ONNX Spec

ONNX

ONNX Model Zoo

Single stack

ONNX RUNTIME

Other compliant runtimes

Network visualization

Visual DL

NETRON

ONNX Open Governance

- ONNX Standard joined LF AI Foundation, Nov 2019
  - Multiple vendors
  - High level steering committee
  - Special Interest Groups (SIGs)
    - Architecture & Infra
    - Converters
    - Operators
    - Models & Tutorials
  - Working groups – e.g. training, pipelines
- Other ecosystem components not part of LFAI (as yet)
  - Converter projects
  - ONNX Runtime
  - Other runtimes

https://lfai.foundation/press-release/2019/11/14/lf-ai-welcomes-onnx/

ONNX Missing Pieces

– ONNX

- Operator / converter coverage
  - e.g. TensorFlow coverage
- Image processing
  - Support for basic resize, crop
  - No support for reading image directly (like tf.image.decode_jpeg)
- String processing
- Comprehensive benchmarks

– ONNX-ML

- Types
  - datetime
- Operators
  - String processing / NLP – e.g. tokenization
  - Hashing
  - Clustering models
- Specific exporters
  - Apache Spark ML – python only
  - Very basic tokenization in sklearn
  - No support for Keras tokenizer
- Combining frameworks
  - Still ad-hoc, requires custom code

Summary

ONNX

- Linux Foundation AI open governance
- Active project; growing rapidly
- Performant - deep learning ops, GPU support
- ONNX-ML provides some support for ”traditional” ML and feature processing

- Still relatively new
- Difficult to keep up with framework evolution
- Still work required for feature processing and other data types (strings, datetime, etc)
- Limited image & text pre-processing

Conclusion

- Open standard for serialization and deployment of deep learning pipelines
  - True portability across languages, frameworks, runtimes and versions
  - Execution environment independent of the producer
  - One execution stack
- Solves a significant pain point for the deployment of DL pipelines in a fully open manner

- However there are risks
  - ONNX still relatively young
  - Operator / framework coverage
  - Limitations of a standard

Get involved - it's open source, open governance!

https://onnx.ai/

Thank you

codait.org

twitter.com/codait_org

twitter.com/MLnick

github.com/CODAIT

developer.ibm.com

Model Asset Exchange

https://ibm.biz/model-exchange

Data Asset Exchange

https://ibm.biz/data-exchange

Sign up for IBM Cloud

https://ibm.biz/Bdqk32