Introduction to ONNX

Open Neural Network Exchange

Objectives

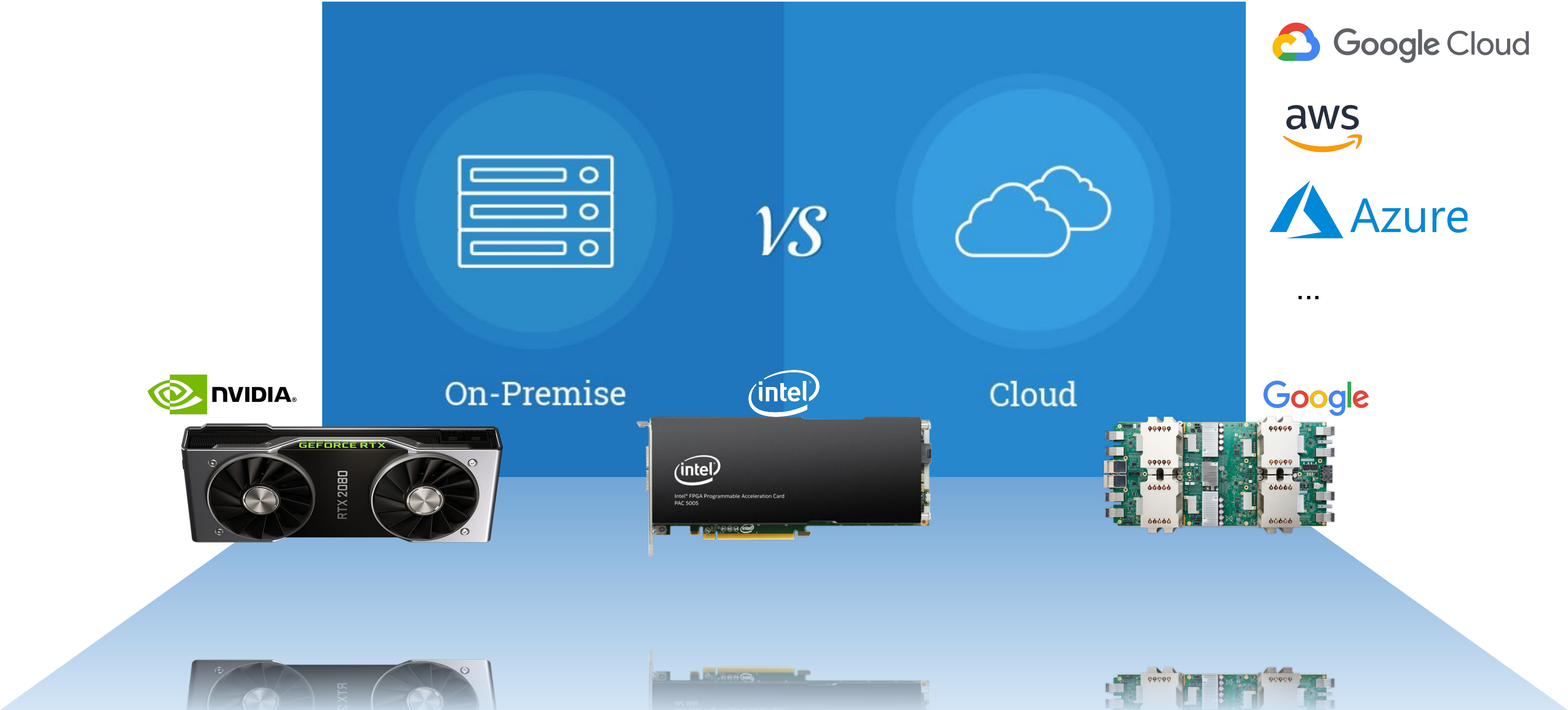
- Challenges with Machine Learning;
- Open Neural Network Exchange
- ONNX Specification;
- ONNX Runtime;
- ONNX Ecosystem;
- ONNX Demos:
  - Create & use a new ONNX model;
  - Use an existing ONNX model;
- Where to find more information?

Challenges #1: ML Framework

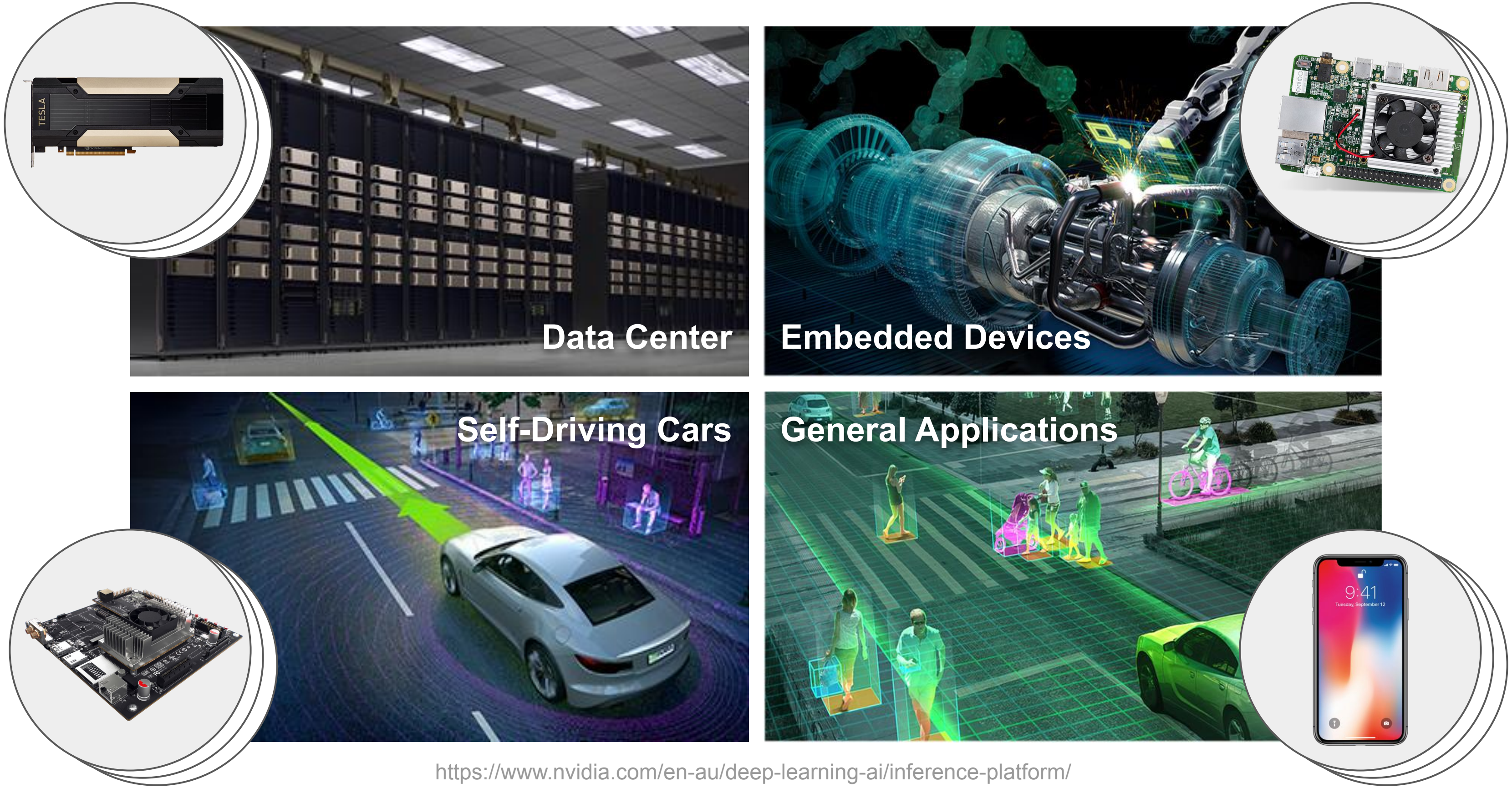
Caffe2
TensorFlow
K
mxnet
PyTorch
scikit
learn
XGBoost
...

https://planet-lean.com/wp-content/uploads/2019/01/Art-Smalley-on-his-new-book-Four-types-of-problems.jpg

Challenges #2: Select Training Infrastructure

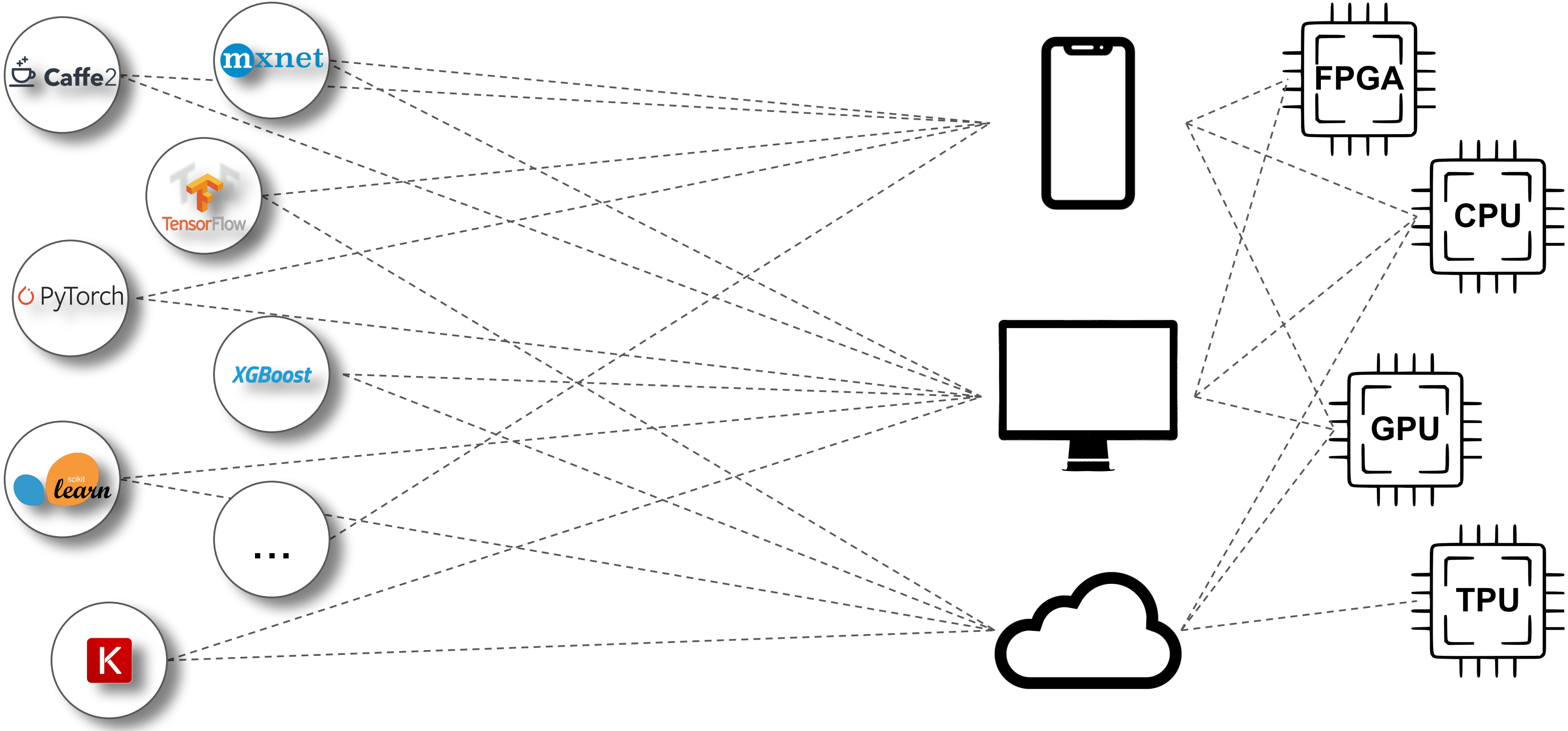


Challenges #3: Select Inference Infrastructure

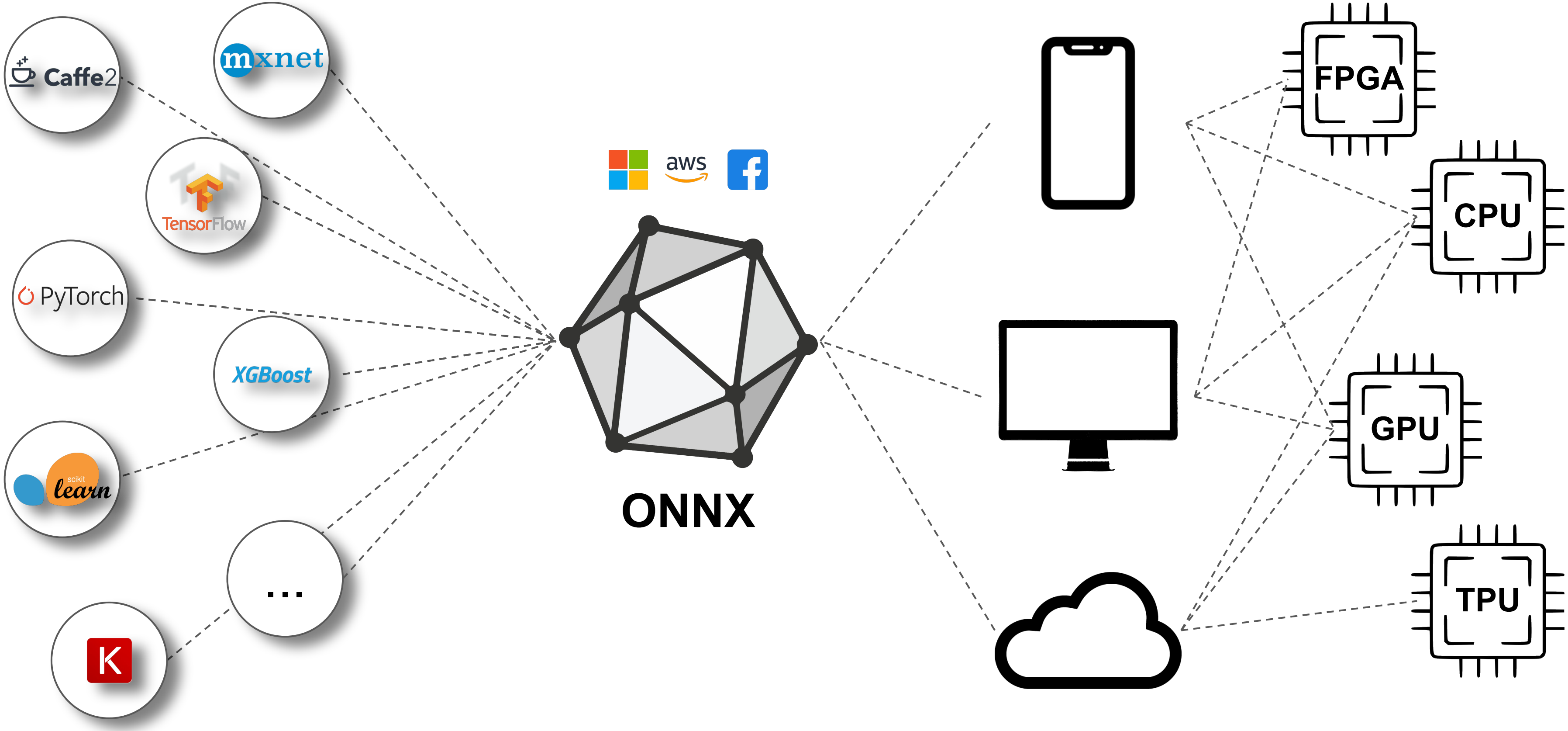


https://www.nvidia.com/en-au/deep-learning-ai/inference-platform/

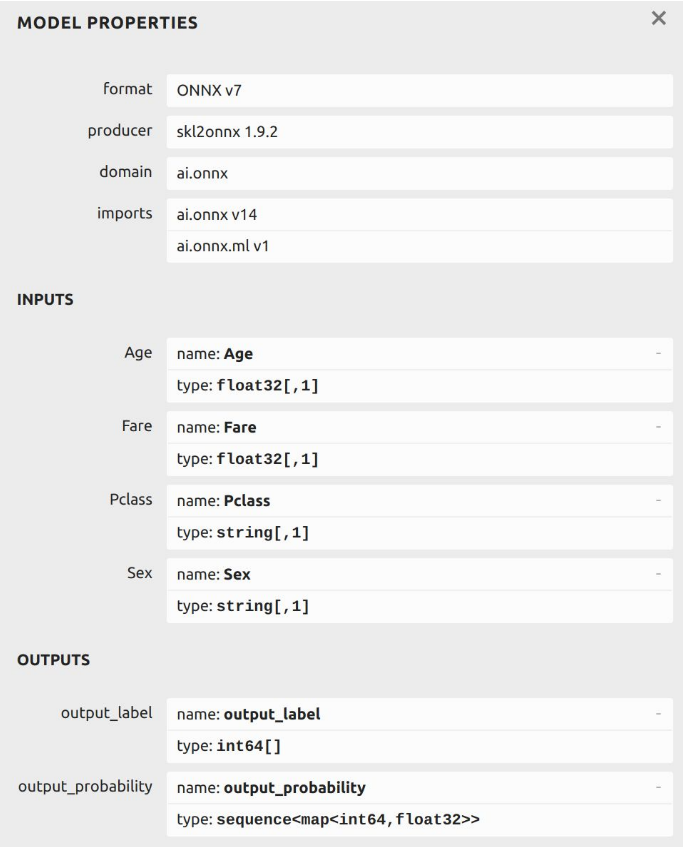
Challenges #4: Summary



Solution: Open Neural Network Exchange



ONNX Specification



MODEL PROPERTIES

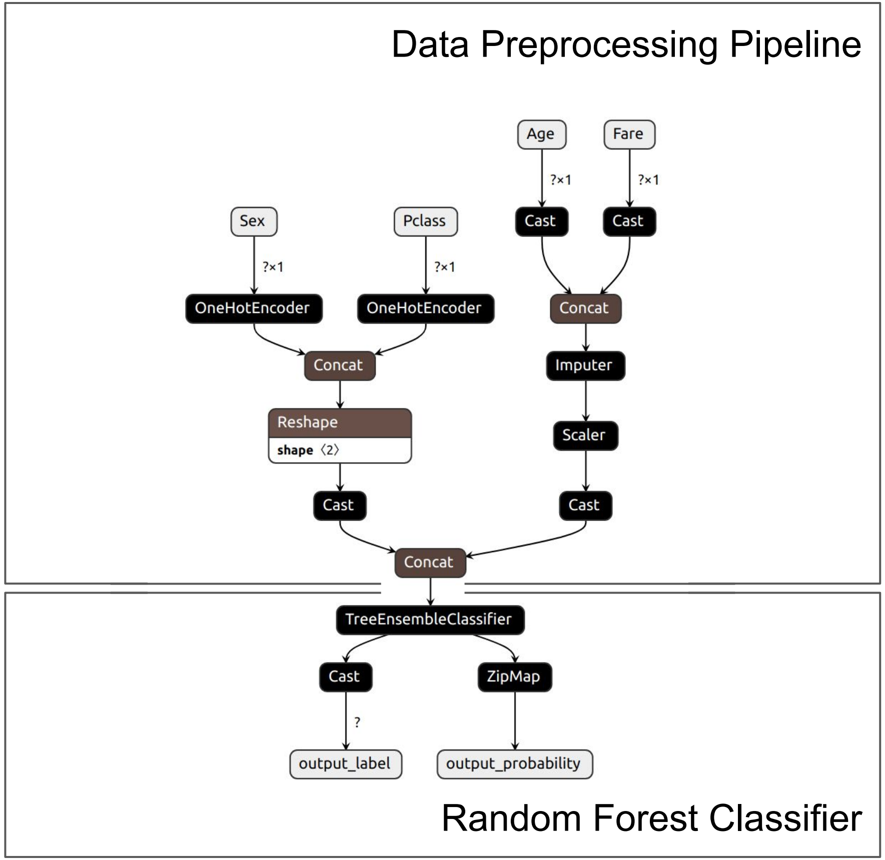
format	ONNX v7
producer	skl2onnx 1.9.2
domain	ai.onnx
imports	ai.onnx v14 ai.onnx.ml v1

INPUTS

Age	name: **Age** type: `float32[,1]`
Fare	name: **Fare** type: `float32[,1]`
Pclass	name: **Pclass** type: `string[,1]`
Sex	name: **Sex** type: `string[,1]`

OUTPUTS

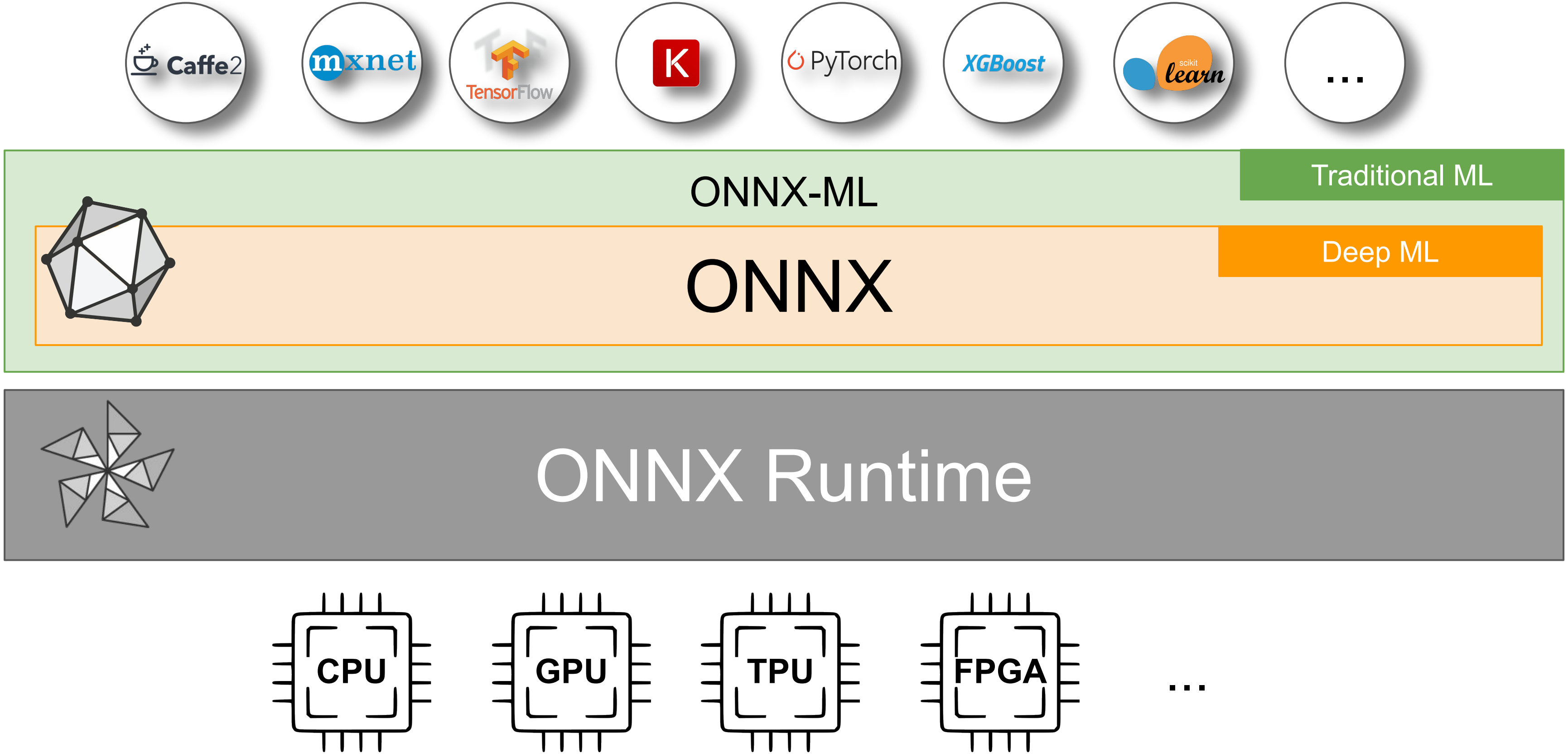
output_label	name: **output_label** type: `int64[]`
output_probability	name: **output_probability** type: `sequence<map<int64,float32>>`



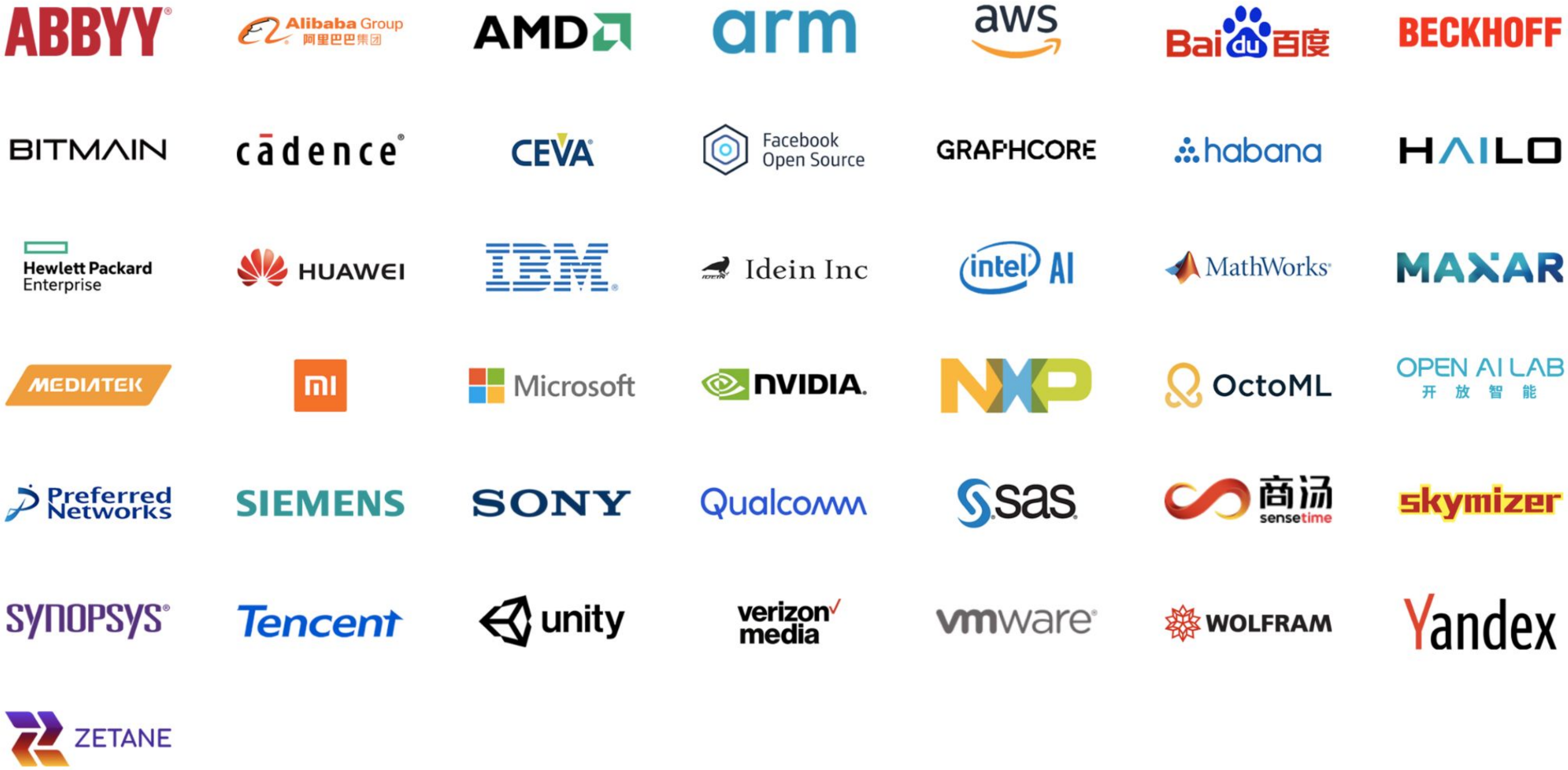
Data Preprocessing Pipeline

Random Forest Classifier

ONNX Runtime

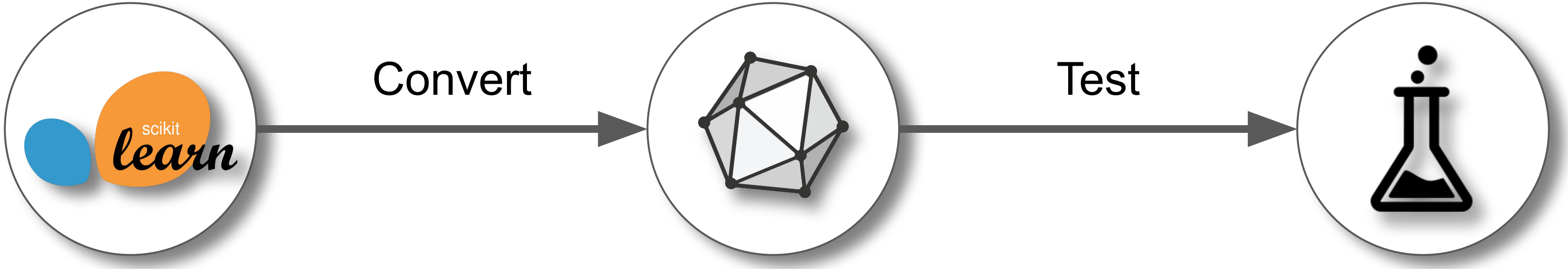


ONNX Ecosystem



https://onnx.ai/

Demo: Create & use a new ONNX model



Convert

Test

Create Random Forest Classifier

ONNX Format

ONNX Inference

Demo: Create & use a new ONNX model (#1)

```python
from utils import create_preprocessor

from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline

# Creates a data preprocessing pipeline.
preprocessor = create_preprocessor(dataset)

# Defines the model as a pipeline which combines a data preprocessor and
# a classifier.
model = Pipeline(steps=[
   ('precprocessor', preprocessor),
   ('classifier', RandomForestClassifier(random_state=42))
])

# Trains the model.
model.fit(train.X, train.y)

# Uses the model for scoring.
model.predict(test.X)
```

Create SKLearn Random Forest Classifier

Demo: Create & use a new ONNX model (#2)

```python
from utils import get_onnx_input_type
from skl2onnx import convert_sklearn

# Creates input type using dataset schema.
initial_type = get_onnx_input_type(train)
print(initial_type)

# Converts the model to the ONNX format.
onnx_model = convert_sklearn(model, initial_types=initial_type)

# Serializes the ONNX model to the file.
with open('model.onnx', "wb") as f:
   f.write(onnx_model.SerializeToString())
```

Convert and Save ONNX Model

```
[('Age', FloatTensorType(shape=[None, 1])),
('Fare', FloatTensorType(shape=[None, 1])),
('Pclass', StringTensorType(shape=[None, 1])),
('Sex', StringTensorType(shape=[None, 1]))]
```

initial_type

Demo: Create & use a new ONNX model (#3)

```python
from utils import get_onnx_input_data
import onnxruntime as rt

input_data = get_onnx_input_data(test)
sess = rt.InferenceSession('model.onnx')
pred, _ = sess.run(None, input_data)
```

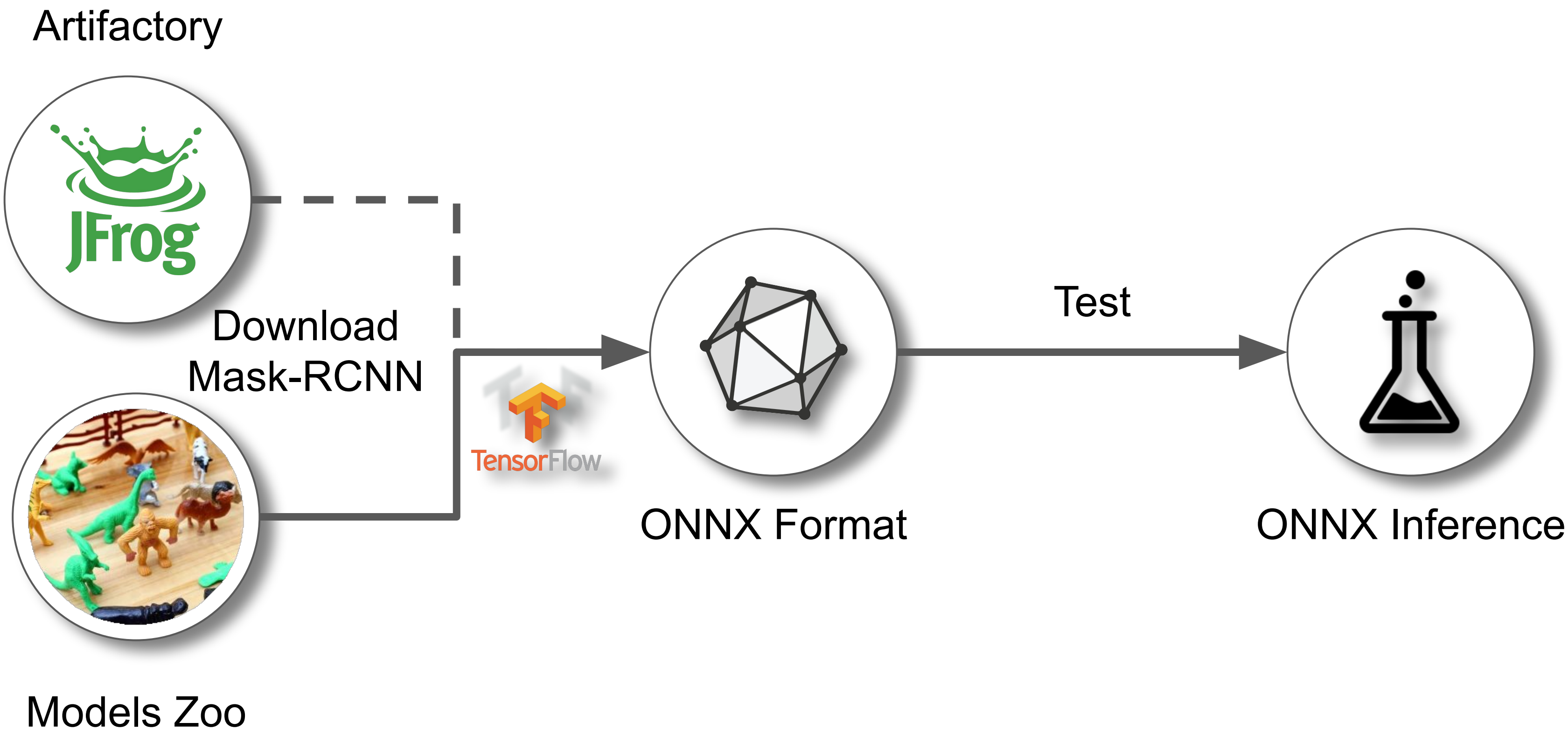
Load and Use ONNX Model

```
      Age       Fare  Pclass     Sex
296  47.0  30.500000       1    male
682  25.0  41.579201       2    male
535  69.0  14.500000       3    male
644  56.0  35.500000       1    male
623  57.0  12.350000       2    male
..    ...        ...     ...     ...
377  19.0   7.775000       3    male
140  22.0   7.750000       3  female
173  56.0  30.695801       1    male
6    54.0  51.862499       1    male
731  28.5  16.100000       3    male
```

Input Data Transformation

```
{'Age': array([
        [47.  ], [25.  ], [69.  ], [56.  ], 57.  ], ...], dtype=float32),
'Fare': array([
        [ 30.5   ], [ 41.5792], [ 14.5   ], [ 35.5   ], [ 12.35  ], ...], dtype=float32),
'Pclass': array([
        ['1'], ['2'], ['3'], ['1'],  ['2'], ...], dtype=object),
'Sex': array([
        ['male'], ['male'], ['male'], ['male'], ...], dtype=object)}
```

Demo: Use an existing ONNX model



Demo: Use an existing ONNX model (#1)

Get Pre-trained Mask-RCNN

```
!wget -O tmp/maskrcnn.onnx
https://github.com/onnx/models/raw/master/vision/object_detection_segmentation/mask-rcnn/model/MaskRCNN-10.onnx
```

Define Pre/Post Process Functions

```
def preprocess(image):
```

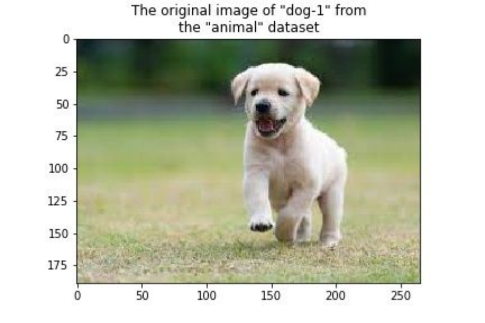
Convert the original image to Mask-RCNN input

```
def postprocess(image, classes, boxes, labels, scores, masks, score_threshold):
```

Take Mask-RCNN predictions and project to the original image...

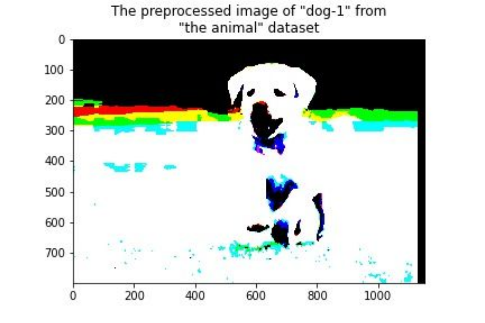
Demo: Use an existing ONNX model (#2)

```
orig_image = load_image(...)
```



Load The Original Image

```
pre_image = preprocess(orig_image)
```



Preprocess input for the Mask-RCNN input

Demo: Use an existing ONNX model (#3)

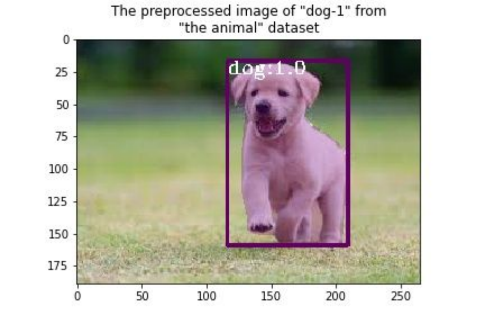
```
import onnxruntime as rt
sess = rt.InferenceSession('maskrcnn.onnx')

f = open('coco_classes.txt')
classes = f.read().split(‘,’)
```

Read ONNX model and COCO classes

```
boxes, labels, scores, masks = sess.run(
   None, {sess.get_inputs()[0].name: pre_image}
)
```

Make Prediction

```
post_image, _ = postprocess( orig_image, classes, boxes, labels, scores, masks, score_threshold)
```



Show Result

Where to find more information?

ONNX Home: https://onnx.ai/

ONNX GitHub: https://github.com/onnx

ONNX Model Zoo: https://github.com/onnx/models

ONNX Runtime: https://onnxruntime.ai/

Open Neural Network Exchange (ONNX) in the enterprise: how Microsoft scales ML: https://www.youtube.com/watch?v=aHk7iUZDIlk

Netron UI: https://netron.app/

Netron GitHub: https://github.com/lutzroeder/netron

Thank You