PyTorch

An open source deep learning platform that provides a seamless path from research prototyping to production deployment.

Scott Sievert

PyTorch

An open source deep learning platform that provides a seamless path from **research prototyping** to production deployment.

Scott Sievert

PyTorch 1.0

An open source deep learning platform that provides **a seamless path** from research prototyping **to production deployment**.

Blog post

Scott Sievert

Outline

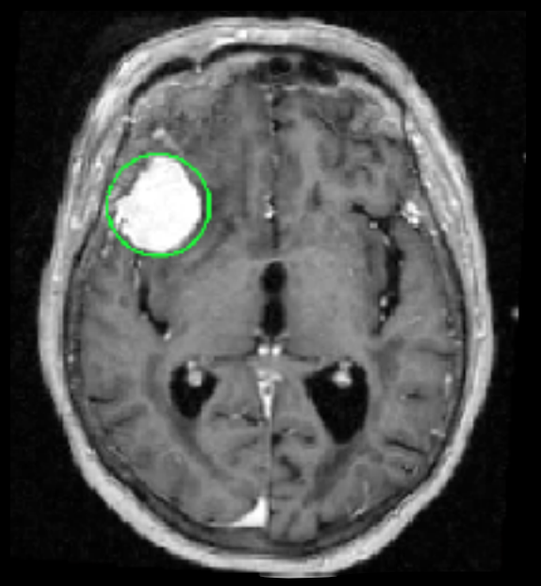
What’s machine learning, and what software is required?

Explanation and demo on features of PyTorch

What’s the enabling technology?

https://github.com/stsievert/talks

Self driving cars



Helping doctors

Siri/Alexa

https://github.com/stsievert/talks

What's the goal of machine learning?

Estimator

No cancer

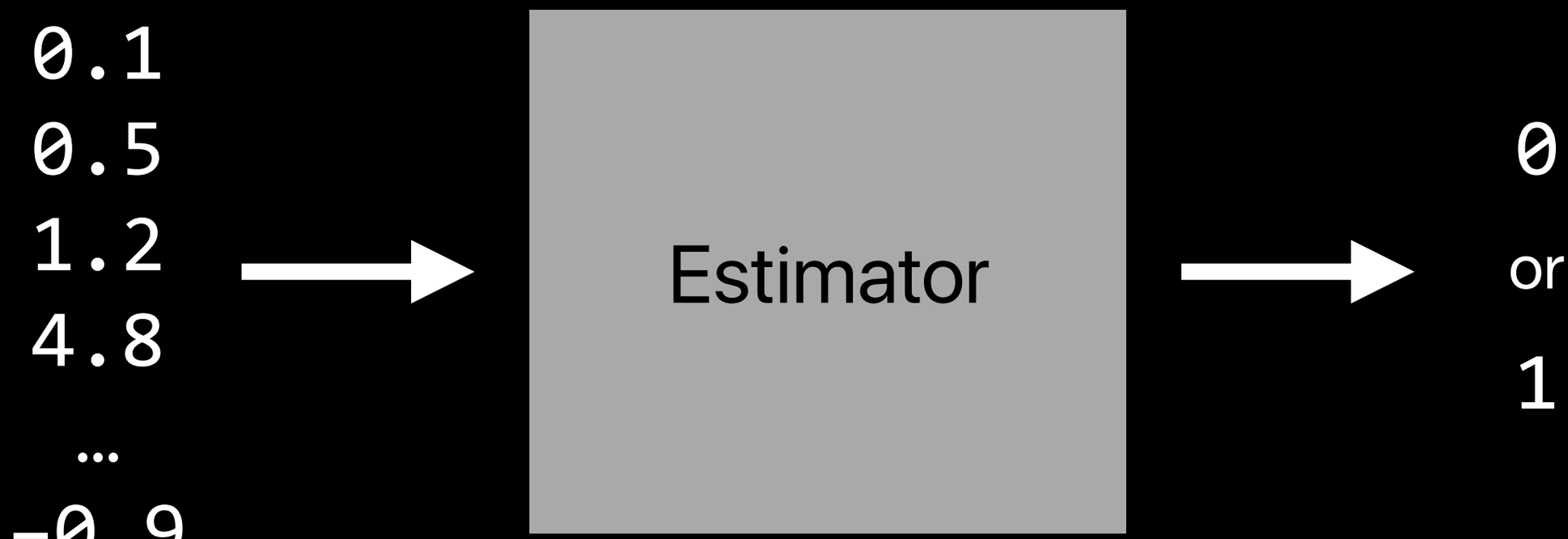
or

Yes cancer

https://github.com/stsievert/talks

What's the goal of machine learning?

To make this
a "good" approximation
for *unseen* data



https://github.com/stsievert/talks

How is the estimator created?

The same way humans learn: by example.

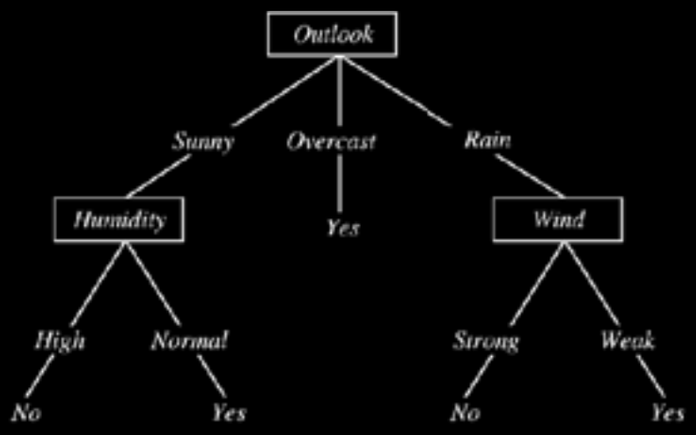
Training data

(many examples)

No cancer

Has cancer

Untrained estimator



Training

Predictor

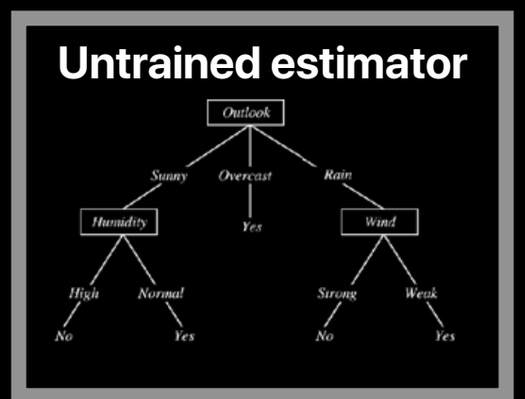
No cancer

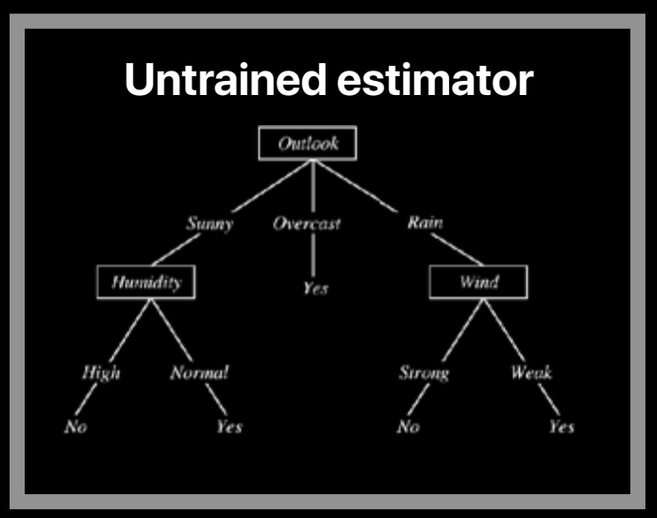
https://github.com/stsievert/talks

What will I talk about?

Software required for **Training**, not the math required

...and specifically not software required for





Training

```sklearn
from sklearn.linear_model import

estimator = LogisticRegression(
    penalty="l1",
    tol=1e-8,
    C=0.1,
    solver="saga",
)
```

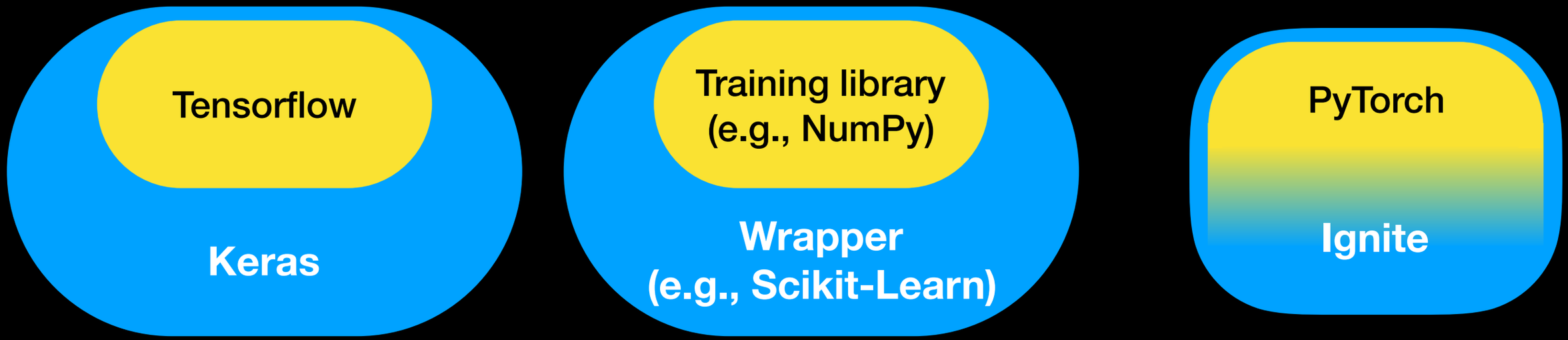
```sklearn
estimator.fit(X_train, y_train)
estimator.score(X_test, y_test)
```

```keras
from keras.models import Seq
from keras.layers import Den

model = Sequential()
model.add(Dense(10, activati

model.compile(
    loss=keras.losses.catego
    optimizer=keras.optimize
)
```

```keras
model.fit(x_train, y_train)
score = model.evaluate(x_tes
```

https://github.com/stsievert/talks

Every library has wrappers to create models easily.



PyTorch requires minimal wrapping

https://github.com/stsievert/talks

François Chollet

@fchollet

Keras creator

Follow

Here is the same dynamic RNN implemented in 4 different frameworks (TensorFlow/Keras, MXNet/Gluon, Chainer, PyTorch). Can you tell which is which?

```
class MyRNN(Block):

    def __init__(self, units=32):
        super(MyRNN, self).__init__()
        self.units = units
        with self.name_scope():
            self.projection_1 = nn.Dense(units=units, activation='tanh')
            self.projection_2 = nn.Dense(units=units, activation='tanh')

    def forward(self, inputs):
        outputs = []
        state = zeros(shape=(inputs.shape[0], self.units))
        for t in range(inputs.shape[1]):
            x = inputs[:, t, :]
            h = self.projection_1(x)
            y = h + self.projection_2(state)
            state = y
            outputs.append(y)
        return concat(*outputs, dim=1)
```

```
class MyRNN(Model):

    def __init__(self, units=32):
        super(MyRNN, self).__init__()
        self.units = units
        self.projection_1 = layers.Dense(units=units, activation='tanh')
        self.projection_2 = layers.Dense(units=units, activation='tanh')

    def call(self, inputs):
        outputs = []
        state = zeros(shape=(inputs.shape[0], self.units))
        for t in range(inputs.shape[1]):
            x = inputs[:, t, :]
            h = self.projection_1(x)
            y = h + self.projection_2(state)
            state = y
            outputs.append(y)
        return concatenate(outputs, axis=1)
```

```
class MyRNN(Module):

    def __init__(self, units=32):
        super(MyRNN, self).__init__()
        self.units = units
        self.projection_1 = layers.Linear(in_features=None, out_features=units)
        self.projection_2 = layers.Linear(in_features=None, out_features=units)

    def forward(self, inputs):
        outputs = []
        state = zeros(shape=(inputs.shape[0], self.units))
        for t in range(inputs.shape[1]):
            x = inputs[:, t, :]
            h = tanh(self.projection_1(x))
            y = h + tanh(self.projection_2(state))
            state = y
            outputs.append(y)
        return stack(outputs, dim=1)
```

```
class MyRNN(Chain):

    def __init__(self, units=32):
        super(MyRNN, self).__init__()
        self.units = units
        with self.init_scope():
            self.projection_1 = layers.Linear(in_size=None, out_size=units)
            self.projection_2 = layers.Linear(in_size=None, out_size=units)

    def forward(self, inputs):
        outputs = []
        state = zeros(shape=(inputs.shape[0], self.units))
        for t in range(inputs.shape[1]):
            x = inputs[:, t, :]
            h = tanh(self.projection_1(x))
            y = h + tanh(self.projection_2(state))
            state = y
            outputs.append(y)
        return stack(outputs, axis=1)
```

9:03 AM - 16 Oct 2018

514 Retweets **1,572** Likes

© 2018 Twitter About
Privacy policy Co

https://github.com/stsievert/talks

I’m going to focus on the training libraries

https://github.com/stsievert/talks

Why should you care?

Training libraries determine....

- Speed
- Memory
- Debugging
- Adoption (StackOverflow, blog posts, etc)
- Performance monitoring
- Integration with other libraries

https://github.com/stsievert/talks

Demo

https://github.com/stsievert/talks

Benefits

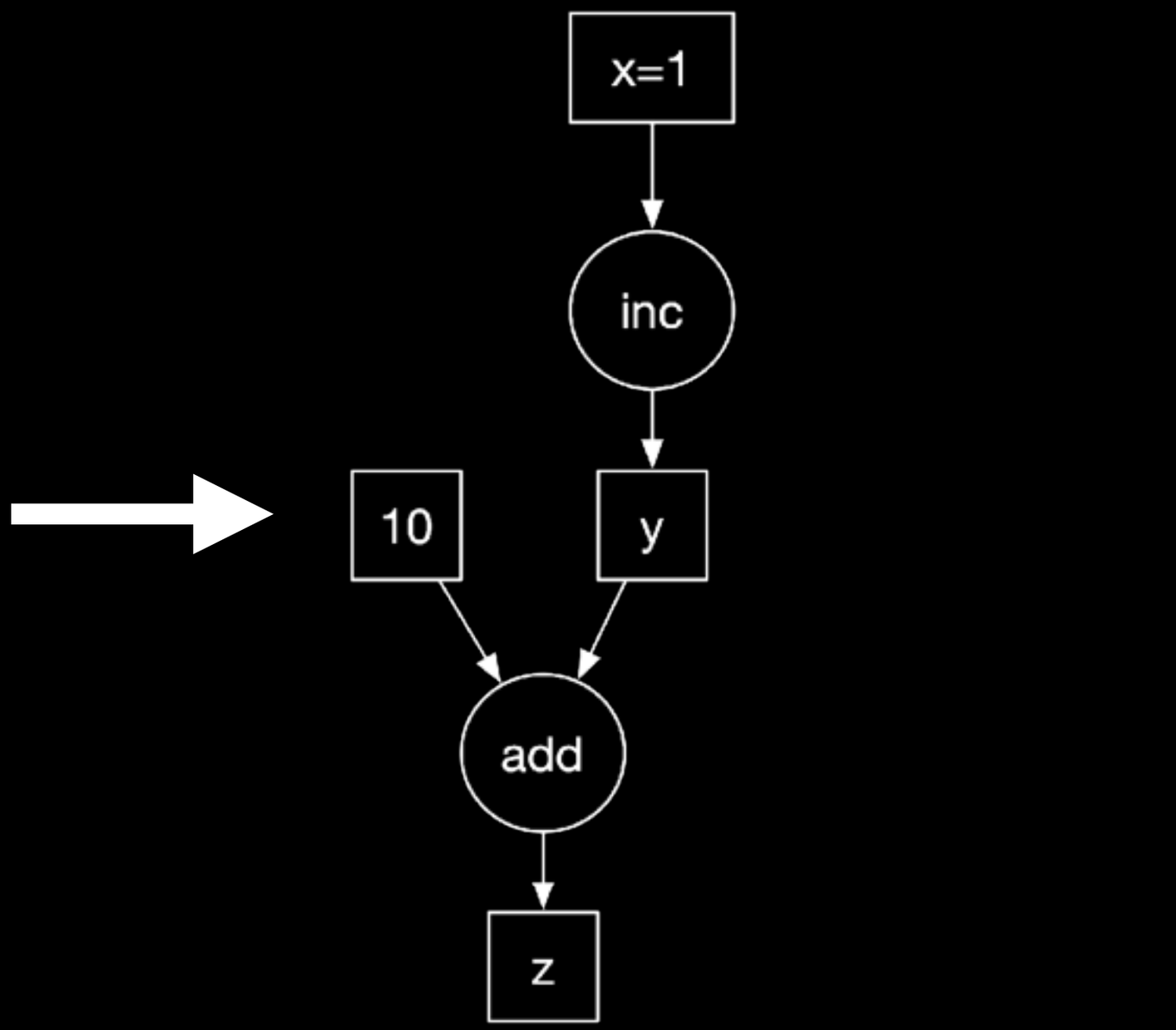
- Easy debugging
- Easy NumPy integration
- Python control flow works

What's the enabling feature behind these benefits?

https://github.com/stsievert/talks

A **graph** describes the
required computation and communication
for a given program

```python
def inc(i):
    return i + 1

def add(a, b):
    return a + b

x = 1
y = inc(x)
z = add(y, 10)
```



https://github.com/stsievert/talks

Graph choices

	Dynamic	**Static**
Execution model	Define by running	Define then run
...which means	Functions run when expected	Functions run once for graph creation
Example	PyTorch	Tensorflow

https://github.com/stsievert/talks

What impact does graph choice have?

- Control flow (e.g., `tf.while_loop`, `tf.cond`)
- Debugging
- Library integration
- Deployment

https://github.com/stsievert/talks

PyTorch

Originally, a dynamic graph system

Now, has static graph:

```
@torch.jit.script
def RNN(h, x, W_h, U_h, W_y, b_h, b_y):
    y = []
    for t in range(x.size(0)):
        h = torch.tanh(x[t] @ W_h + h @ U_h + b_h)
        y += [torch.tanh(h @ W_y + b_y)]
        if t % 10 == 0:
            print("stats: ", h.mean(), h.var())
    return torch.stack(y), h
```

https://github.com/stsievert/talks

Tensorflow

Originally, a static graph system.

Now, has dynamic graph through "eager execution"

Eager: Distributed Execution #14129

Open asimshankar opened this issue on Oct 31, 2017 · 1 comment · Jump to bottom

Eager: Random seeds #14134

Open asimshankar opened this issue on Oct 31, 2017 · 1 comment · Jump to bottom

Eager: Variable item-assignment #14132

Open asimshankar opened this issue on Oct 31, 2017 · 3 comments · Jump to bottom

Eager: CPU Performance/Operation Overheads #14130

Closed asimshankar opened this issue on Oct 31, 2017 · 2 comments · Jump to bottom

https://github.com/stsievert/talks

Deployment

Branch: master ▾ pytorch / caffe2 /

Caffe2 and PyTorch join forces to create a Research + Production platform PyTorch 1.0

Posted May 02, 2018

https://caffe2.ai/blog/2018/05/02/Caffe2_PyTorch_1_0.html

Branch: master ▾ pytorch / torch / onnx /

Open Neural Net Exchange (ONNX)

https://github.com/stsievert/talks

ONNX

OPEN NEURAL NETWORK EXCHANGE FORMAT

The new open ecosystem for interchangeable AI models

Caffe2 · Chainer · Cognitive Toolkit · mxnet · PyTorch · PaddlePaddle · MATLAB

K · ML · learn · dmlc XGBoost · LibSVM

Compilers

intel AI · skymizer · tvm

NETRON · Visual DL

NVIDIA · Qualcomm · BITMAIN

Tencent · vespa · Windows

SYNOPSYS · ONNX · CEVA

https://github.com/stsievert/talks

Thanks!

Questions?

PyTorch	Tensorflow
Performance	Performance
Debugging	Debugging
Integration with other libraries	Integration with other libraries
Deployment	Deployment
Community	Community

https://github.com/stsievert/talks