Deep learning with Pytorch

Mansura Habiba
Advisory Software Engineer
IBM Talent Management Solutions

Courtesy: PyLadies Dublin

Agenda

01

Introduction to pytorch

02

Building blocks of pytorch

03

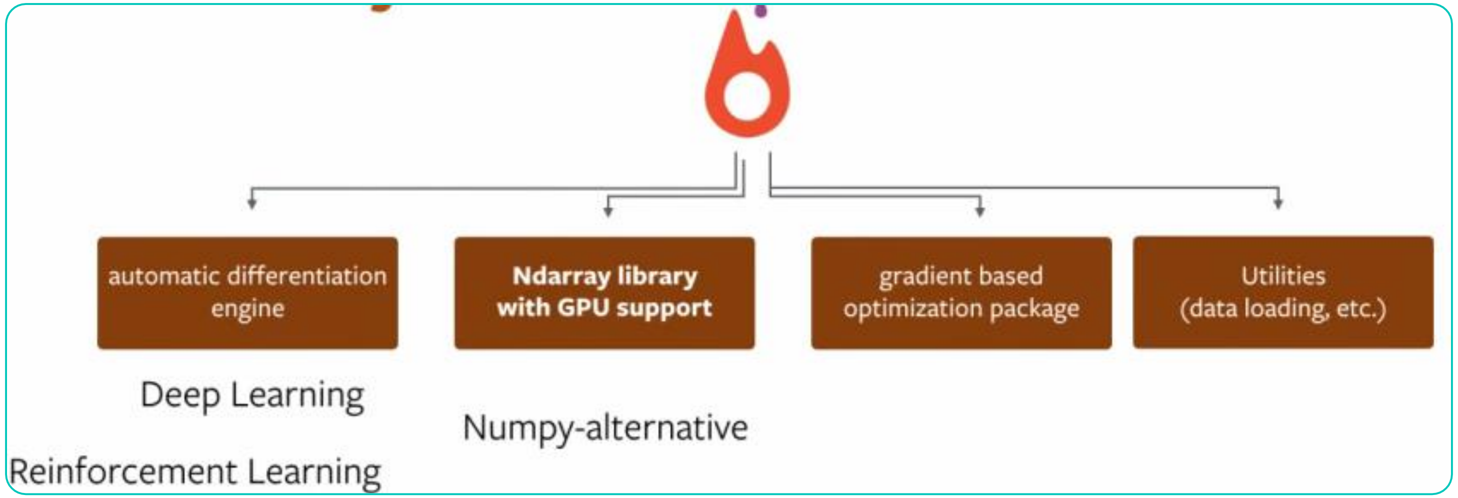
Deep learning with pytorch

- Linear regression
- Sequence generation using NN

04

pytorch and production

Introduction to Pytorch



- Lua torch
- C –libraries
- Automatic Differentiation Engine
- Nd Library with GPU
- Open-sourced by Facebook 2017

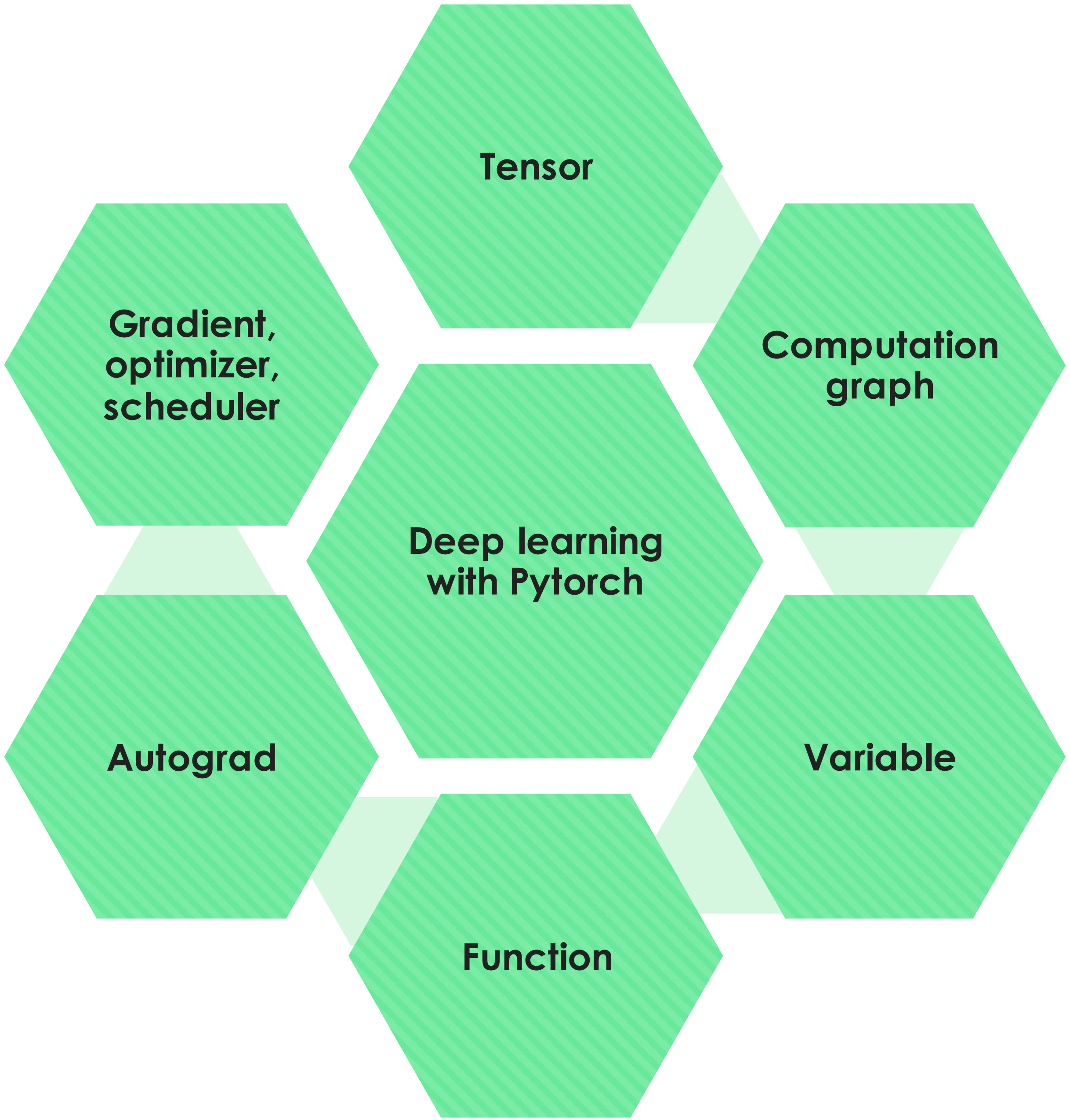
Installation of pytorch

- Anaconda
- Git repo
- Docker Image
- Helm Kubernetes

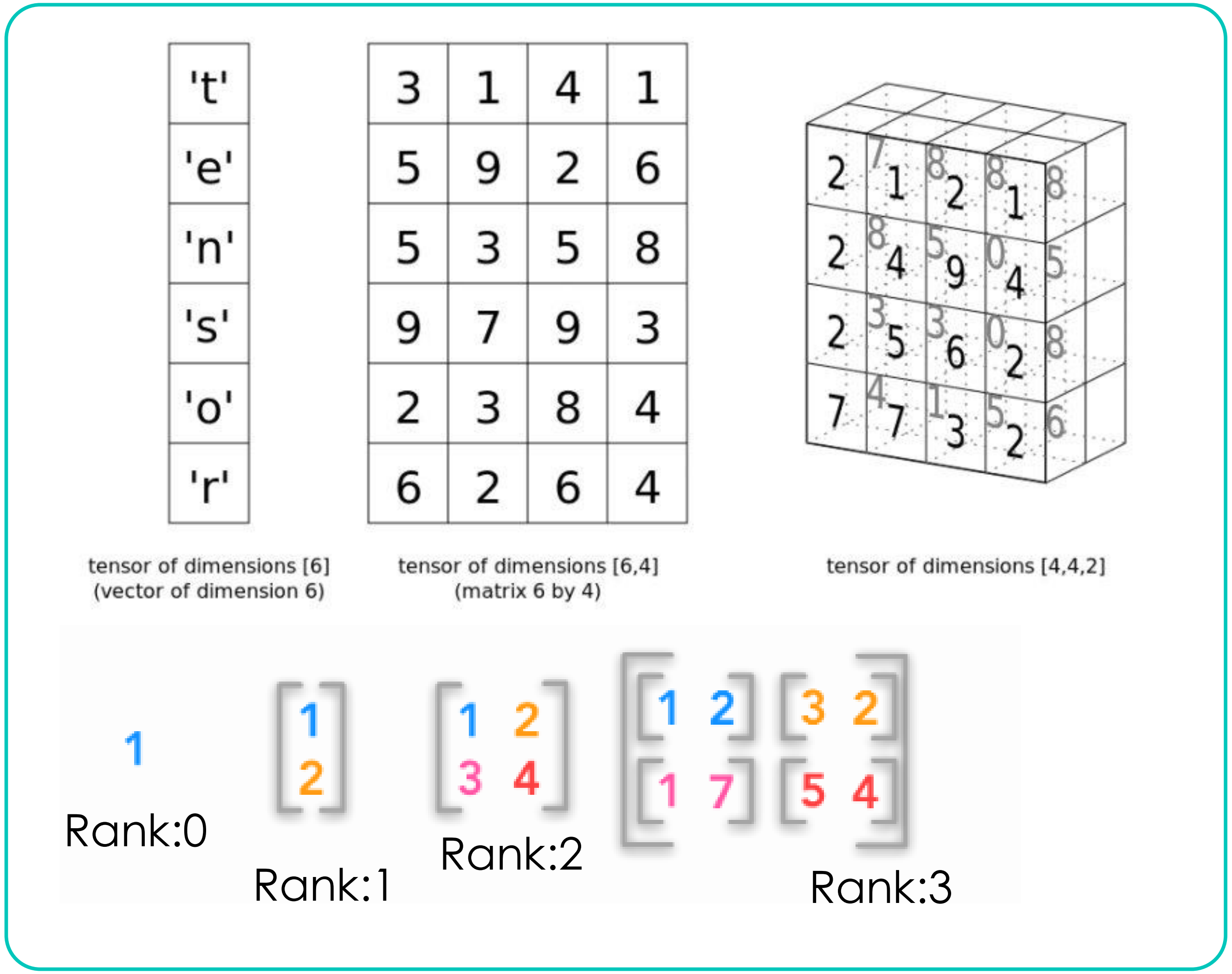
Setup Docker Container

Install Docker	**docker pull pytorch/pytorch**
Run the docker image	**docker run -it --name pytorch1 -v current_dir_including_ipynb_files:/workspace -p 5000:8888 -p 5001:6006 pytorch/pytorch**
Check the libraries	**pip freeze**
Install jupyter	**pip install jupyter**
Run Jupyter	**jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root &**

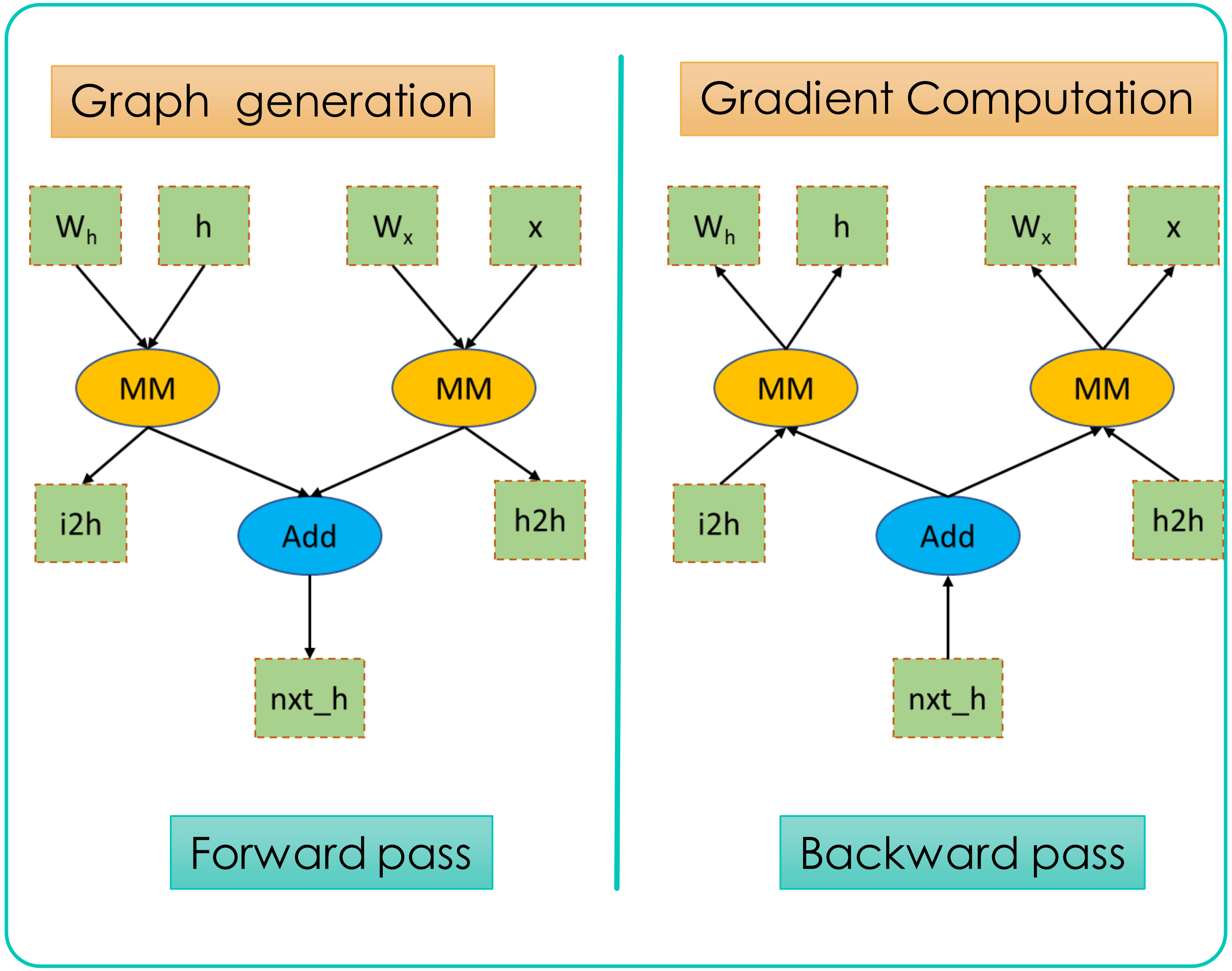
Building blocks for deep learning in Pytorch



Brief introduction Tensor



Dynamic Computation Graph



- Why dynamic?
- retain_grah = true

Computational graph toolkit

Declarative toolkit

```python
import tensorflow as tf
import numpy as np

trX = np.linspace(-1, 1, 101)
trY = 2 * trX + np.random.randn(*trX.shape) * 0.33

X = tf.placeholder("float")
Y = tf.placeholder("float")

def model(X, w):
    return tf.multiply(X, w)

w = tf.Variable(0.0, name="weights")
y_model = model(X, w)

cost = tf.square(Y - y_model)

train_op = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

with tf.Session() as sess:
    tf.global_variables_initializer().run()

    for i in range(100):
        for (x, y) in zip(trX, trY):
            sess.run(train_op, feed_dict={X: x, Y: y})

    print(sess.run(w))
```

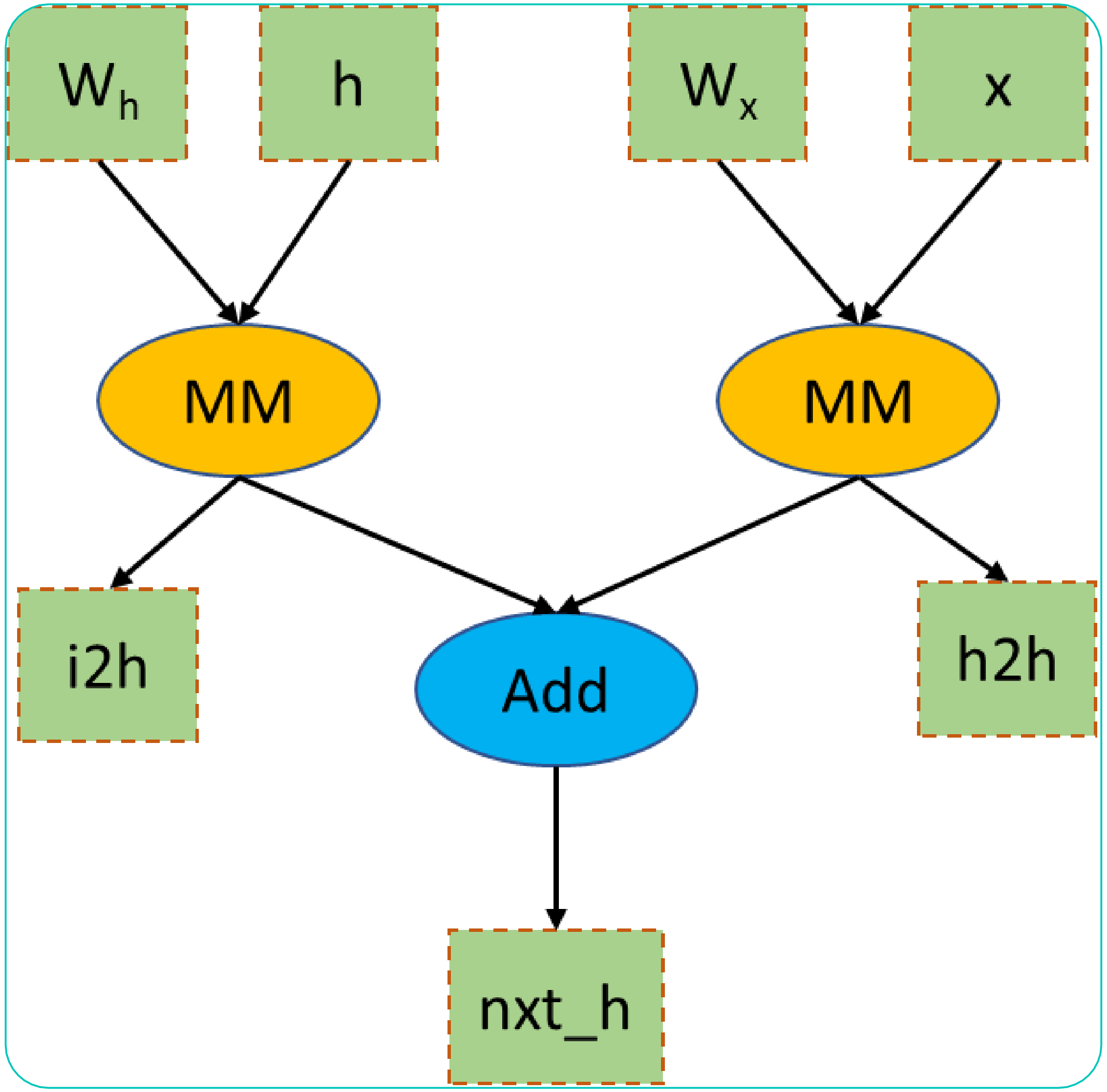
Input / output placeholders

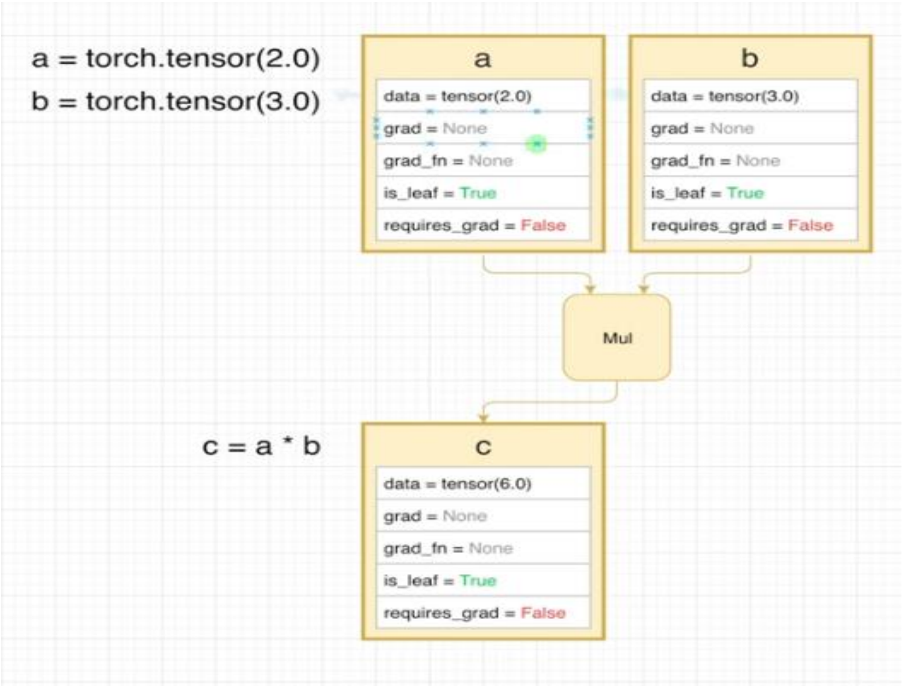
Imperative toolkit

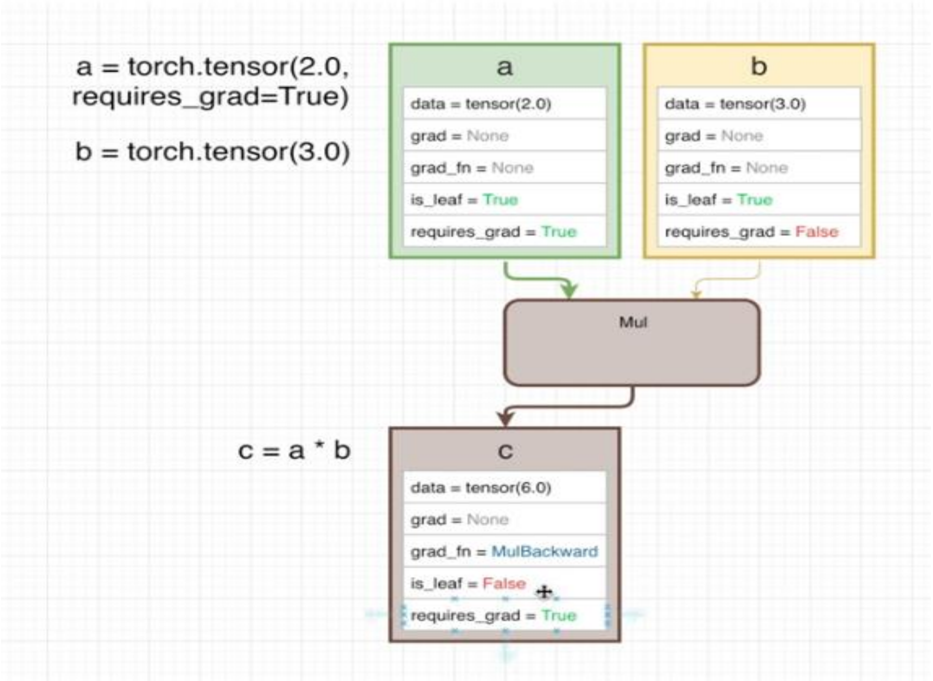
```python
import torch
from torch.autograd import Variable

trX = torch.linspace(-1, 1, 101)
trY = 2 * trX + torch.randn(*trX.size()) * 0.33

w = Variable(trX.new([0.0]), requires_grad=True)

for i in range(100):
  for (x, y) in zip(trX, trY):
    X = Variable(x)
    Y = Variable(y)

    y_model = X * w.expand_as(X)
    cost = (Y - Y_model) ** 2
    cost.backward(torch.ones(*cost.size()))

    w.data = w.data + 0.01 * w.grad.data

  print(w)
```

Variable

- Wrapper around a tensor
- It store history all the operations done on the tensor
- Main purposes are
  - Computation graph specification
  - Accumulation of gradients
  - Facilitate Back propagation, automatic differentiation
- requires_grad = True
- *torch.no_grad()*



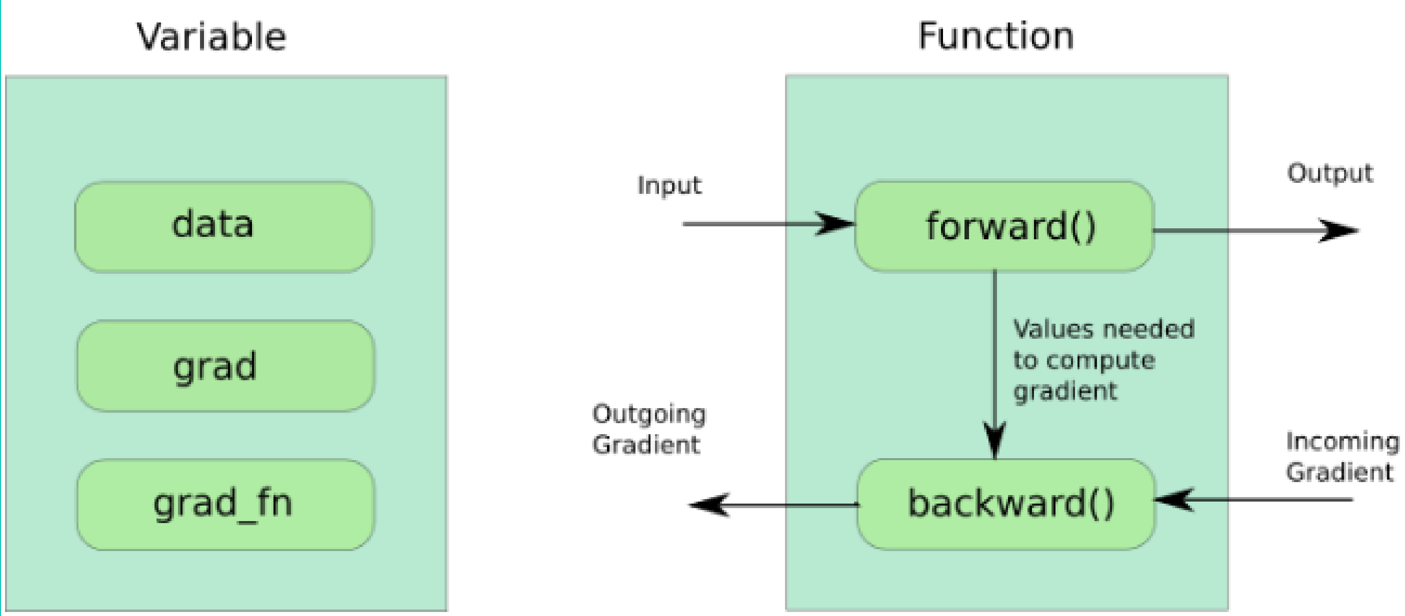




require_grad=False

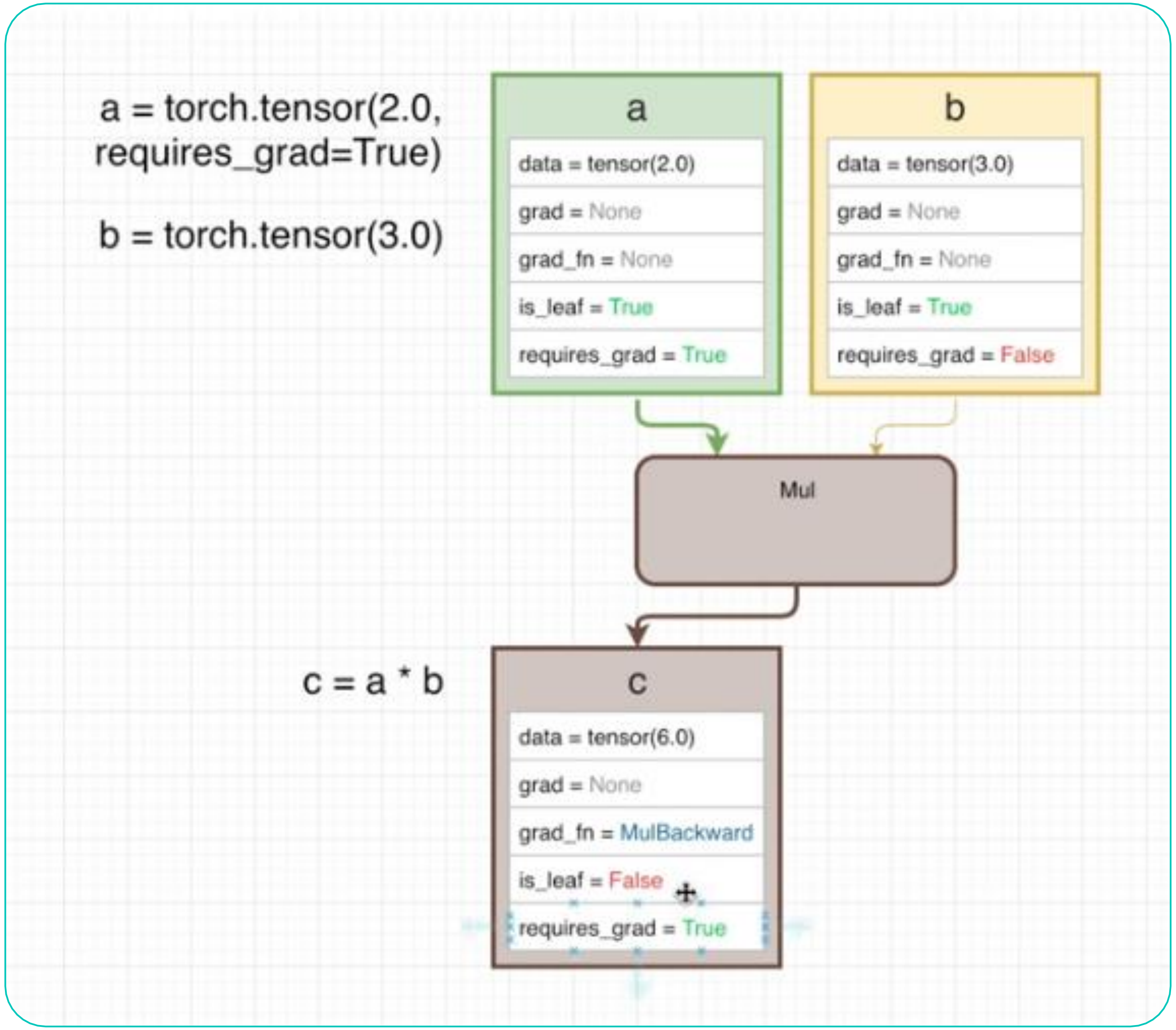
Function

- next_h = i2h+h2h
- grad_fn = AddBackWard

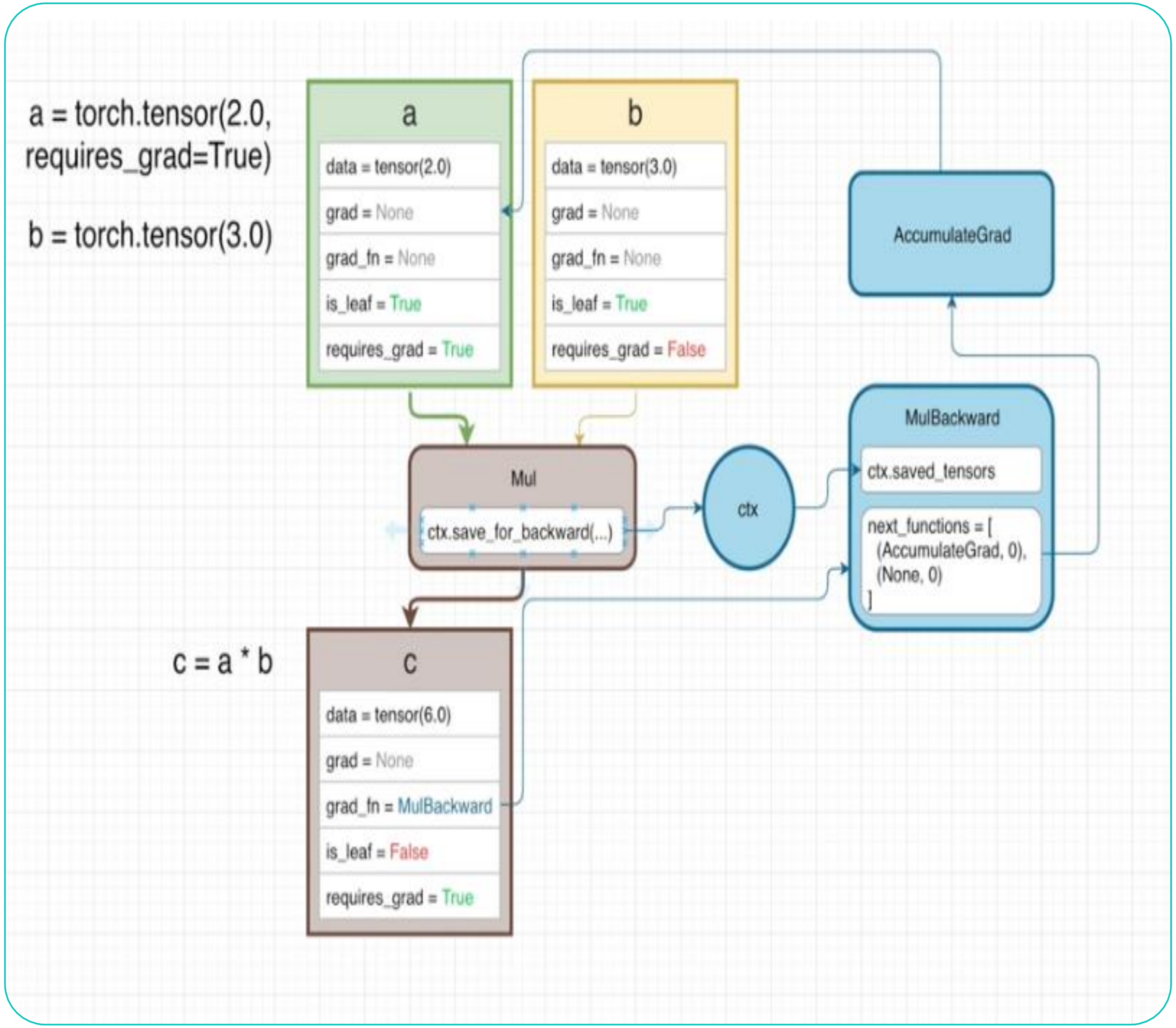


Autograd

- Forward pass
- Computation Graph geenration



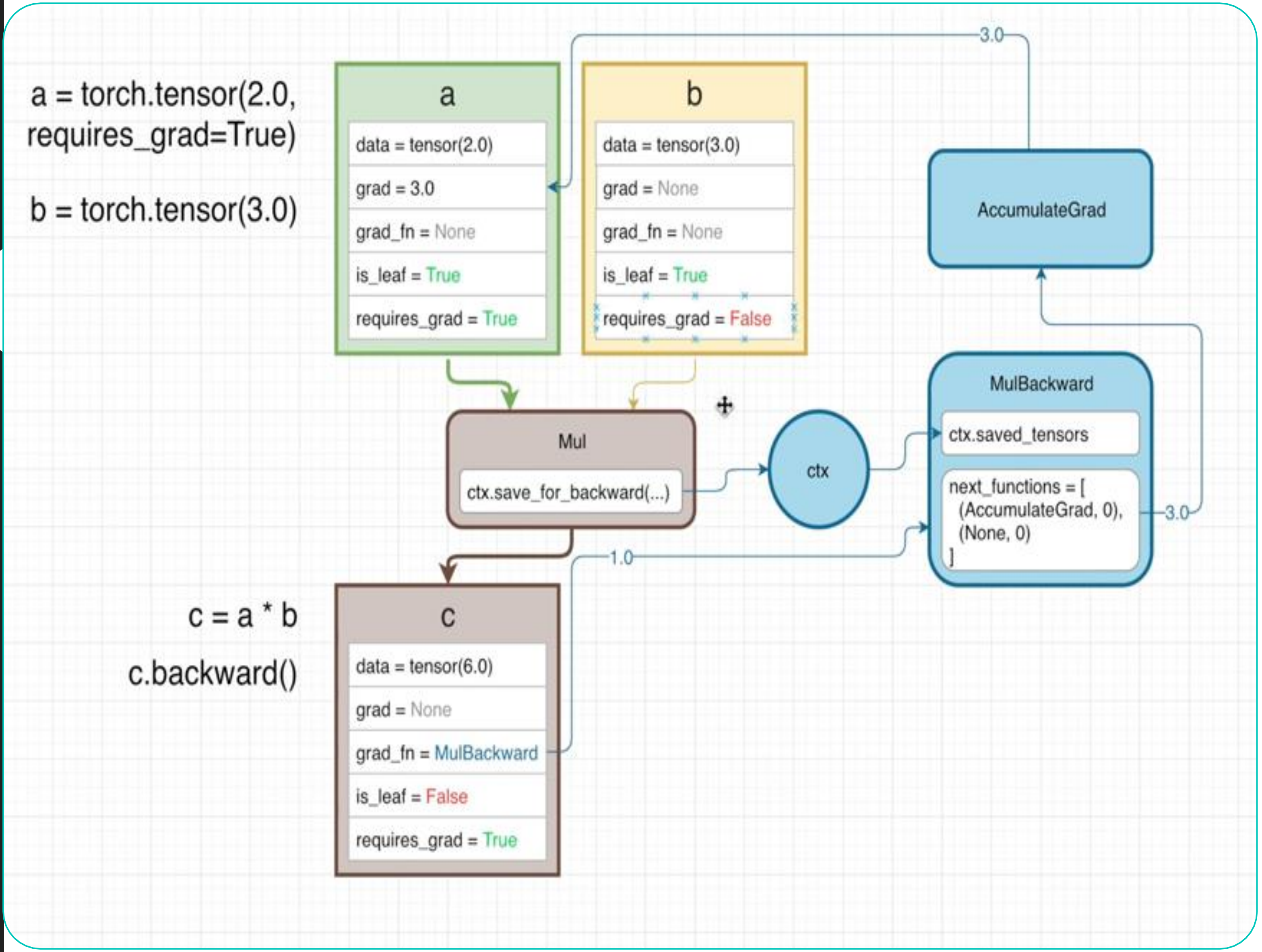
Autograd



- Backward pass
- Computation gradient

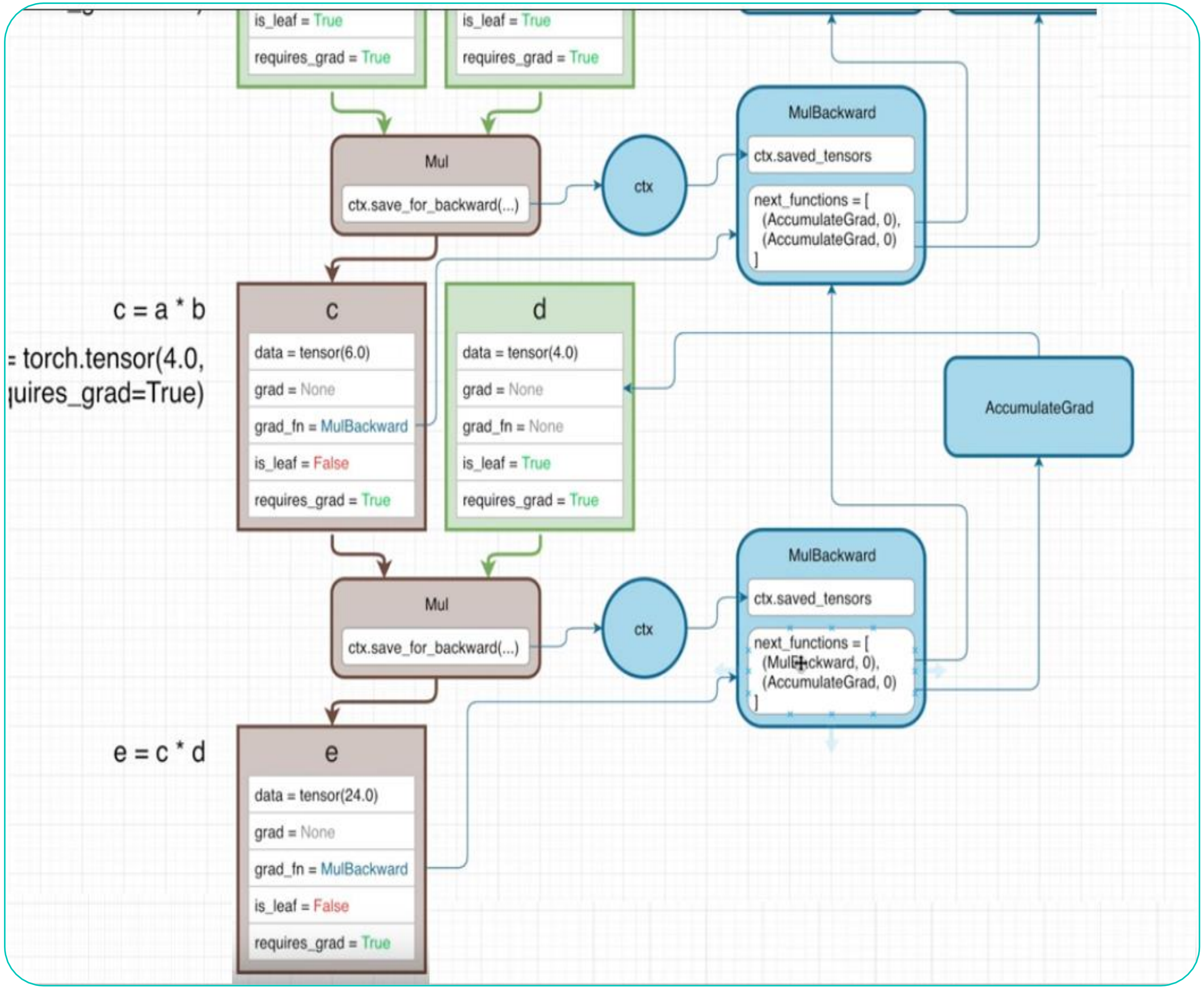
Autograd

- *Variable* also stores the gradient of a scalar quantity (say, loss) with respect to the parameter it holds



Autograd

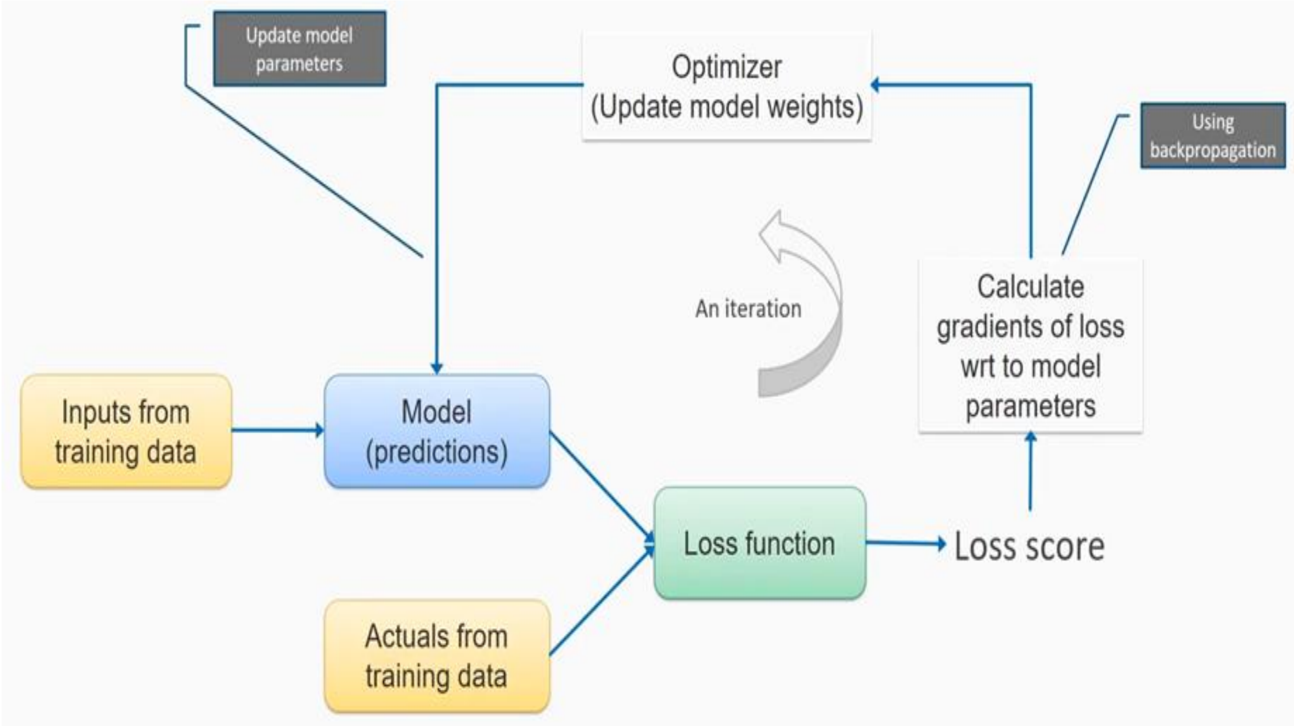
- More deep tree



Pytorch and automatic differentiation – in a nutshell

```
from torch.autograd import Variable
x, prev_h = Variable(torch.randn(1, 10)), Variable(torch.randn(1, 20))
W_h, W_x = Variable(torch.randn(20, 20)), Variable(torch.randn(20, 10))

i2h = torch.matmul(W_x, x.t())
h2h = torch.matmul(W_h, prev_h.t())
(i2h + h2h).tanh().sum().backward()
```

- AD for pytorch written in C++
- Every intermediate result records only the subset of the computation graph that was relevant to their computation
- Support Invalidation and Aliasing




```
for input, target in dataset:
    optimizer.zero_grad()
    output = model(input)
    loss = loss_fn(output, target)
    loss.backward()
    optimizer.step()
```

Role of Optimizer

SGD, Adagrad, RMSProp, LBFGS etc.

Why do we need to call zero_grad() in PyTorch?

- In PyTorch, we need to set the gradients to zero before starting to do backpropragation because PyTorch accumulates the gradients on subsequent backward passes.
- So, the default action is to accumulate the gradients on every loss.backward() call.

```
import torch
from torch.autograd import Variable
import torch.optim as optim

def linear_model(x, W, b):
    return torch.matmul(x, W) + b

data, targets = ...

W = Variable(torch.randn(4, 3), requires_grad=True)
b = Variable(torch.randn(3), requires_grad=True)

optimizer = optim.Adam([W, b])

for sample, target in zip(data, targets):
    # clear out the gradients of all Variables
    # in this optimizer (i.e. W, b)
    optimizer.zero_grad()
    output = linear_model(sample, W, b)
    loss = (output - target) ** 2
    loss.backward()
    optimizer.step()
```

Why Pytorch is fast

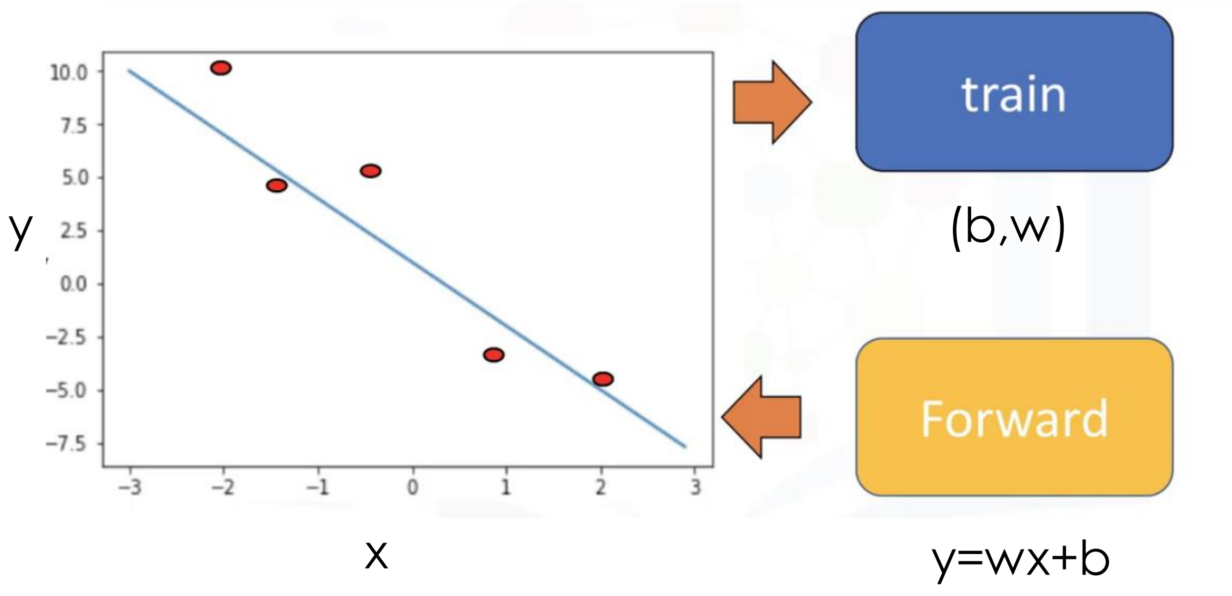
Avoid GIL by JIT

Use Cuda/ GPU

Main Modules of Pytorch

Neural Network Model	Autograd	Optimizer
torch.nn	torch.autograd	torch.optim

Linear regression



LR in General way

- b = -1 , w=2
- y = -1 + 2x
- x =1
- y = -1 + 2(1)
- y = 1

```
import torch

w=torch.tensor(2.0,requires_grad=True)
b=torch.tensor(-1.0,requires_grad=True)

def forward(x):
        y=w*x+b
        return y
x=torch.tensor([[1.0]])

yhat=forward(x)

yhat: tensor([[1.0]])
```

```python
import torch

w=torch.tensor(2.0,requires_grad=True)
b=torch.tensor(-1.0,requires_grad=True)

def forward(x):
        y=w*x+b
        return y
x=torch.tensor([[1.0]])

yhat=forward(x)

yhat: tensor([[1.0]])
```

```python
from torch.nn import  Linear as Linear
```

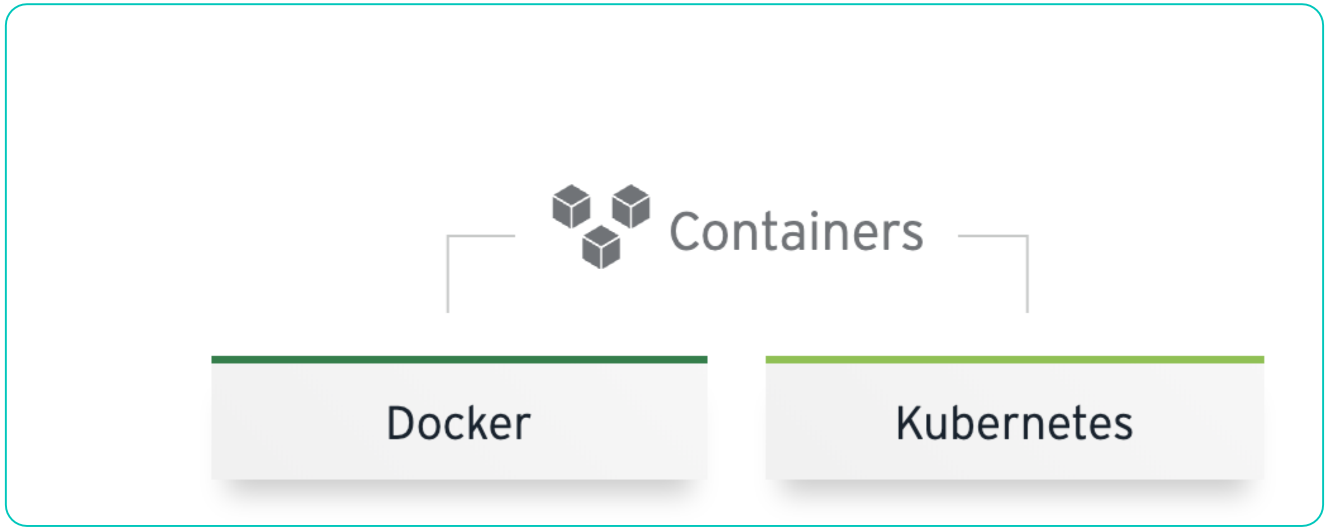
```python
model=Linear(in_features=1,out_features=1)
```

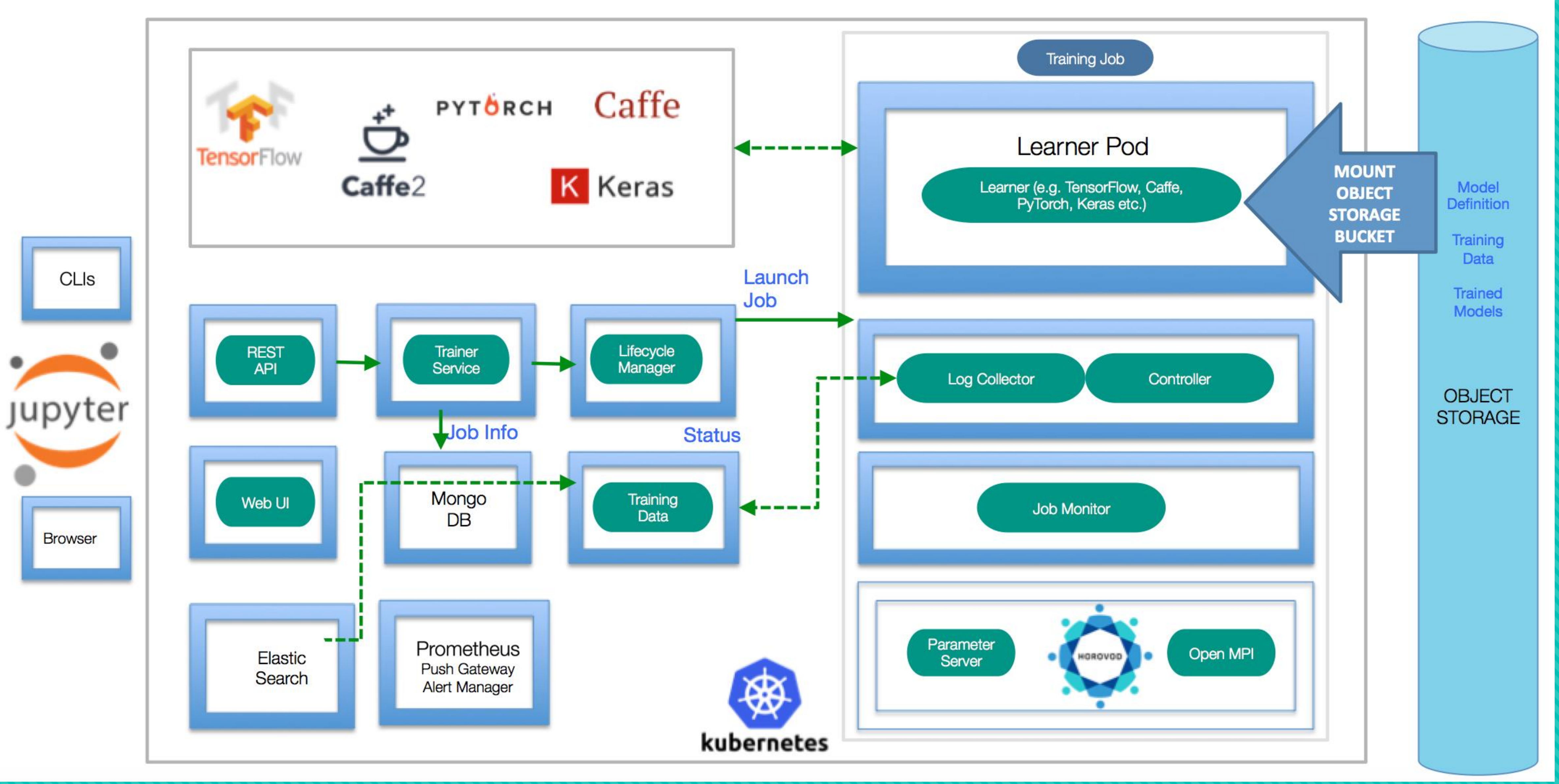
```python
yhat=model (x)
```

LR in Pytorch way

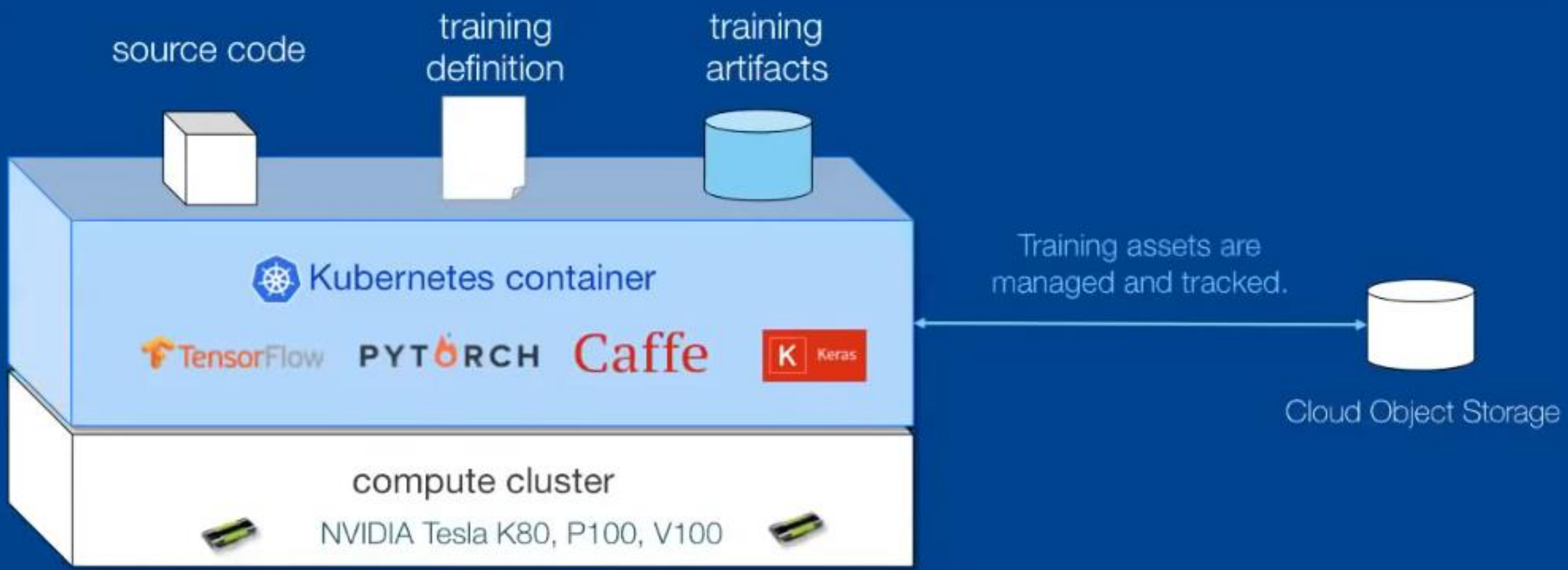
Choices for Production

- Save the model torch.save
- Load the model torch.load

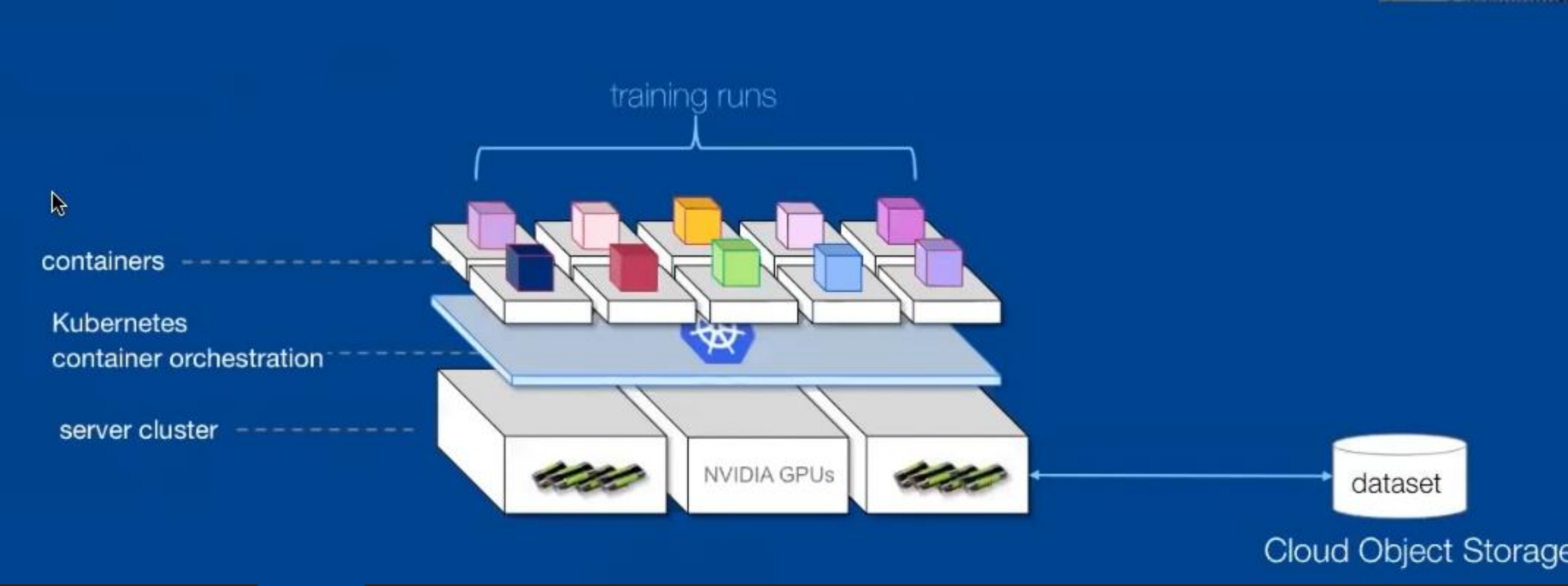




Deep Learning ecosystem for production : IBM FfDL


source code
training definition
training artifacts
Kubernetes container
TensorFlow
PYTORCH
Caffe
Keras
compute cluster
NVIDIA Tesla K80, P100, V100
Training assets are managed and tracked.
Cloud Object Storage

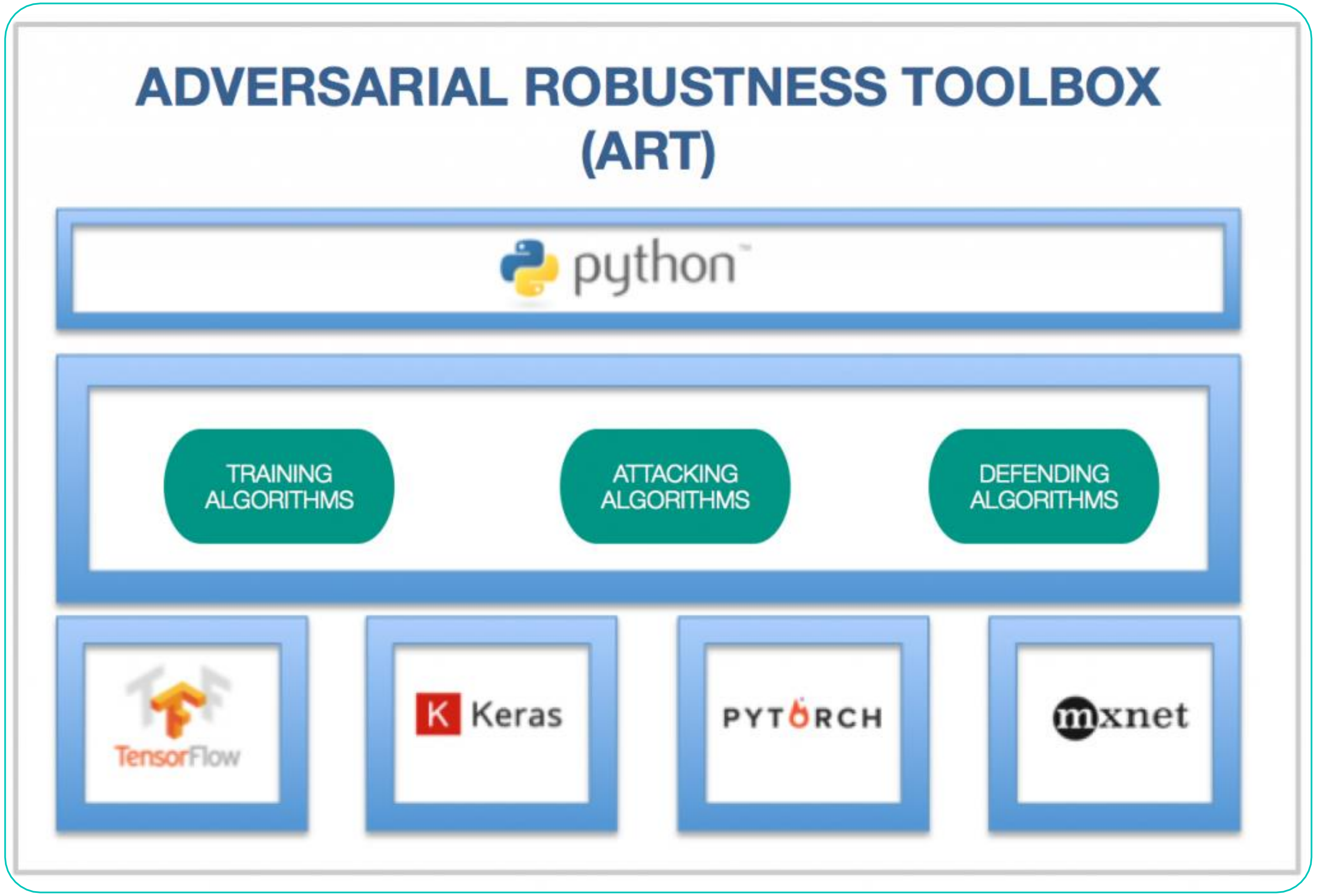
IBM FfDL



Training Model using FfDL

Popular Library of FfDI

ART – (https://github.com/IBM/adversarial-robustness-toolbox)



Start With IBM:FfDL

01

[Getting Stared]

02

[Sample Model]

03

[Pytorch Model]

Reference

- Paul O'Grady - An introduction to PyTorch & Autograd

Learning Materials

Official tutorials: https://pytorch.org/tutorials/

Examples: https://github.com/pytorch/examples

Course : Deep Learning with Python and PyTorch

Playlist : http://deeplizard.com/learn/video/v5cngxo4mlg

Podcast : https://podtail.com/en/podcast/this-week-in-machine-learning-ai-podcast/pytorch-fast-differentiable-dynamic-graphs-in-pyth/

Thank you

Partial Derivative

f = uv +u^2

$$f(u=1, v=2) = uv + u^2$$

$$1(2) + 1^2 = 3$$

$$\frac{\partial f(1,2)}{\partial u}, \frac{\partial f(1,2)}{\partial v}$$

$$\frac{\partial f(u,v)}{\partial u} = \mathrm{v} + 2\mathrm{u}$$

$$\frac{\partial f(u=1, v=2)}{\partial u} = 2 + 2(1)$$

Derivative of u

$$f(u=1, v=2) = uv + u^2$$

$$1(2) + 1^2 = 3$$

$$\frac{\partial f(1,2)}{\partial u}, \frac{\partial f(1,2)}{\partial v}$$

$$\frac{\partial f(u,v)}{\partial v} = \mathrm{u}$$

$$\frac{\partial f(u=1, v=2)}{\partial v} = 1$$

$$= 1$$

Derivative of v