An Introduction to Gluon

Julien Simon
Global Technical Evangelist, AI & Machine Learning, AWS
@julsimon

Apache MXNet and Gluon

Apache MXNet

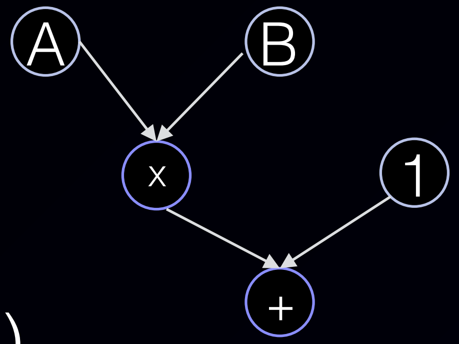
- Open source software library for Deep Learning
- Natively implemented in C++
- Built-in support for many network architectures: FC, CNN, LSTM, etc.
- Symbolic API: Python, Scala, Clojure, R, Julia, Perl, Java (inference only)
- Imperative API: Gluon (Python), with computer vision and natural language processing toolkits

Symbolic execution

‘define then run’

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),

B=np.ones(10)*2)
```



C can share memory with D because C is deleted later

PROS

- More chances for optimization
- Language independent
- E.g. TensorFlow, Theano, Caffe, MXNet

CONS

- Less flexible
- ’Black box’ training

Apache MXNet : *Symbol* API

```
import mxnet as mx

train_iter = mx.io.MNISTIter(shuffle=True)
val_iter = mx.io.MNISTIter(image="./t10k-images-idx3-ubyte", label="./t10k-labels-idx1-ubyte")

data = mx.sym.Variable('data')
data = mx.sym.Flatten(data=data)
fc1  = mx.sym.FullyConnected(data=data, name='fc1', num_hidden=512)
act1 = mx.sym.Activation(data=fc1, name='relu1', act_type="relu")
drop1= mx.sym.Dropout(data=act1,p=0.2)
fc2  = mx.sym.FullyConnected(data=drop1, name='fc2', num_hidden = 256)
act2 = mx.sym.Activation(data=fc2, name='relu2', act_type="relu")
drop2= mx.sym.Dropout(data=act2,p=0.2)
fc3  = mx.sym.FullyConnected(data=drop2, name='fc3', num_hidden=10)
mlp  = mx.sym.SoftmaxOutput(data=fc3, name='softmax’)

mod = mx.mod.Module(mlp, context=mx.cpu(0))
mod.bind(data_shapes=train_iter.provide_data, label_shapes=train_iter.provide_label)
mod.init_params(initializer=mx.init.Xavier())
mod.init_optimizer('adam', optimizer_params=(('learning_rate', 0.1),))

mod.fit(train_iter, eval_data=val_iter, num_epoch=50)
```

Define
a network

Train an
optimized
version

Imperative execution

‘define by run’

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

PROS

- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger).
- E.g. Numpy, PyTorch, Gluon API

CONS

- Harder to optimize

Apache MXNet : Gluon API

```
import mxnet as mx
from mxnet import gluon, autograd, ndarray

train_data = …
test_data = …

net = gluon.nn.Sequential()
with net.name_scope():
    net.add(gluon.nn.Dense(128, activation="relu"))
    net.add(gluon.nn.Dense(64, activation="relu"))
    net.add(gluon.nn.Dense(10))

net.collect_params().initialize(mx.init.Normal(sigma=0.05))
softmax_cross_entropy = gluon.loss.SoftmaxCrossEntropyLoss()
trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate’: 0.1})

epochs = 10
for e in range(epochs):
    for i, (data, label) in enumerate(train_data):
        data = data.as_in_context(mx.cpu()).reshape((-1, 784))
        label = label.as_in_context(mx.cpu())
        with autograd.record():
            output = net(data)
            loss = softmax_cross_entropy(output, label)
            loss.backward()
        trainer.step(data.shape[0])
```

Define a network

Train using the same network

Demo

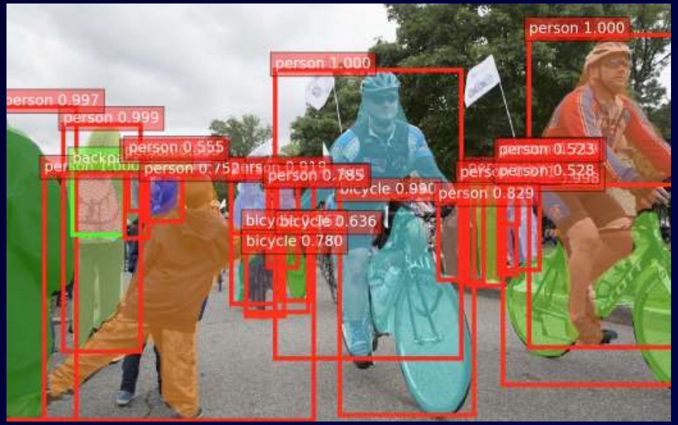
https://gitlab.com/juliensimon/dlnotebooks/blob/master/mxnet/06%20-%20MNIST%20with%20Gluon%20API.ipynb

GluonCV and GluonNLP

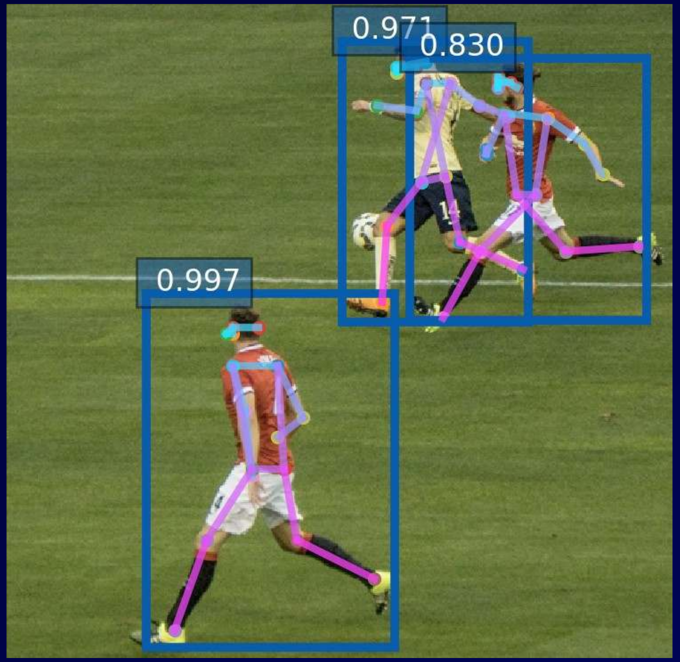
GluonCV

https://gluon-cv.mxnet.io
https://github.com/dmlc/gluon-cv

- State-of-the-art deep learning tools for computer vision
  - Pre-trained models
  - Training and fine-tuning scripts
  - Prototype products, validate new ideas and learn computer vision



- Image classification: 50+ models
- Object detection: Faster RCNN, SSD, Yolo-v3
- Semantic segmentation: FCN, PSP, DeepLab v3
- Instance segmentation: Mask RCNN
- Pose estimation: Simple Pose
- Person re-identification (Market1501 dataset)
- GANs: Wasserstein GAN, Super Resolution GAN, CycleGAN



Demos

gluon-cv/docs/tutorials

GluonNLP

https://gluon-nlp.mxnet.iohttps://github.com/dmlc/gluon-nlp

- State-of-the-art deep learning tools for natural language processing
  - Pre-trained models and embeddings
  - Training and fine-tuning scripts
  - Prototype products, validate new ideas and learn NLP

- Word embeddings: Word2Vec, FastText, GloVE, BERT
- Machine translation: GNMT, Transformer
- Sentiment analysis: TextCNN
- Text classification: FastText
- Language models
- Text generation
- Natural language inference
- Parsing

Demos

Deep Learning on AWS

Amazon SageMaker

- Collect and prepare training data
- Choose and optimize your ML algorithm
- Set up and manage environments for training
- Train and Tune ML Models
- Deploy models in production
- Scale and manage the production environment

AWS Deep Learning AMI
AWS Deep Learning containers*

TensorFlow K Microsoft CNTK ONNX

mxnet GLUON Chainer HOROVOD

Caffe2 PYTORCH torch

Amazon EC2

c
5

p3

* Tensorflow and MXNet only
PyTorch coming soon

aws

Apache MXNet on Amazon SageMaker: a first-class citizen

- Built-in containers for training and prediction
  - Code available on Github: https://github.com/aws/sagemaker-mxnet-container
  - Build it, run it on your own machine, customize it, push it to Amazon ECR, etc.
  - Supported versions: 0.12.1, 1.0.0, 1.1.0, 1.2.1, 1.3.0

- Advanced features
  - Local mode: train on the notebook instance for faster experimentation
  - Script mode: use the same TensorFlow as on your local machine
  - Distributed training: zero setup!
  - Pipe mode: stream large datasets directly from Amazon S3
  - Keras support (tf.keras.* and keras.*)

Demo

Getting started

http://aws.amazon.com/free

https://aws.amazon.com/sagemaker
https://github.com/aws/sagemaker-python-sdk
https://github.com/awslabs/amazon-sagemaker-examples

https://medium.com/@julsimon
https://gitlab.com/juliensimon/dlnotebooks

Thank you!

Julien Simon
Global Technical Evangelist, AI & Machine Learning, AWS
@julsimon