CVPR SEATTLE WASHINGTON JUNE 16-18 2020

CVPR2020 Video Modeling Tutorial

Deployment

Zhi Zhang, AWS, June 2020

aws

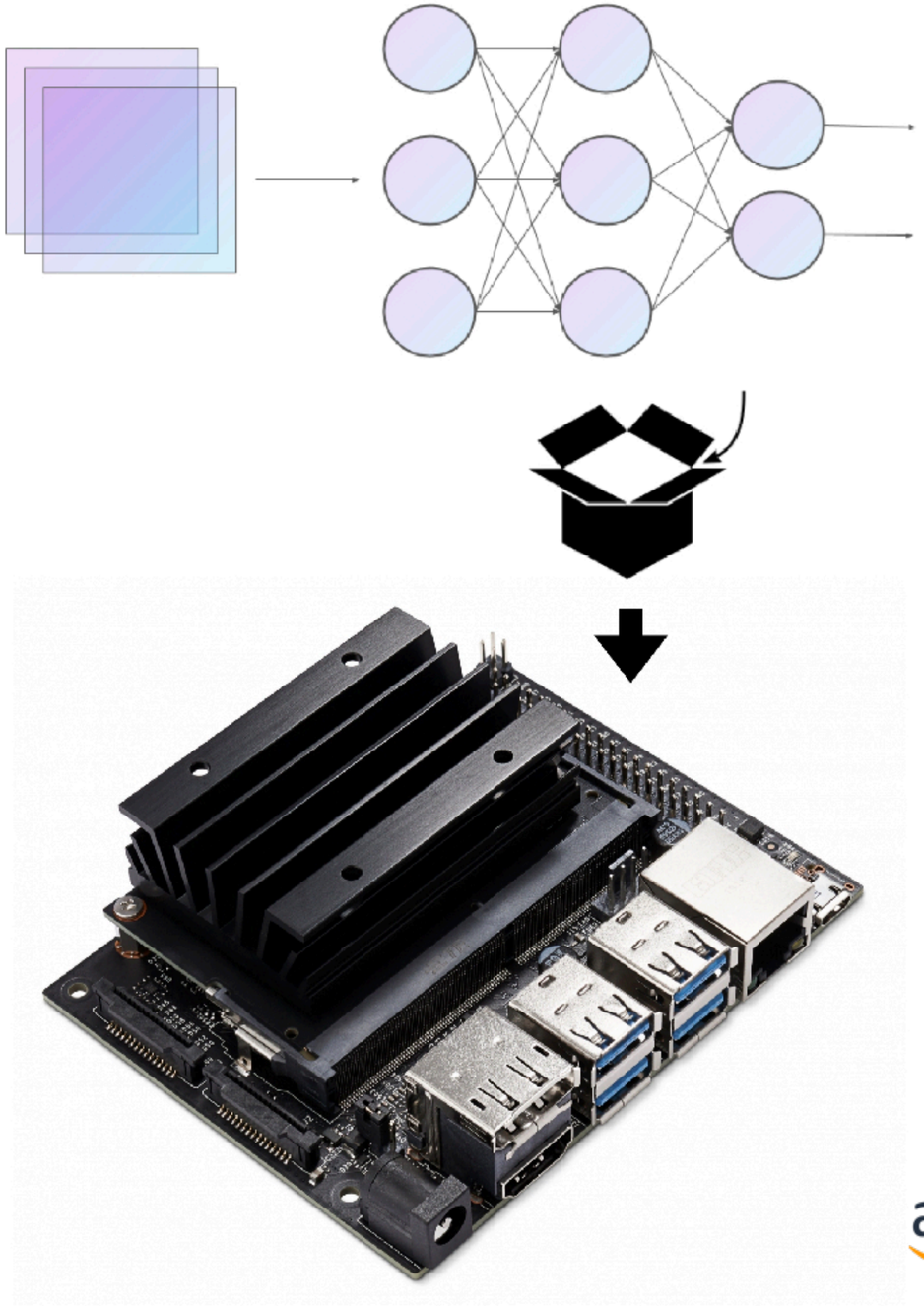
Outline

- Why we take deployment seriously
- Why deployment of deep video models is hard
- How we apply the best practices (notebook walkthrough)
- Things to expect in the near future
- Take-aways

Why deployment

In order to start using a deep learning model for practical workload in commercial products or production service, it needs to be efficiently deployed into production.

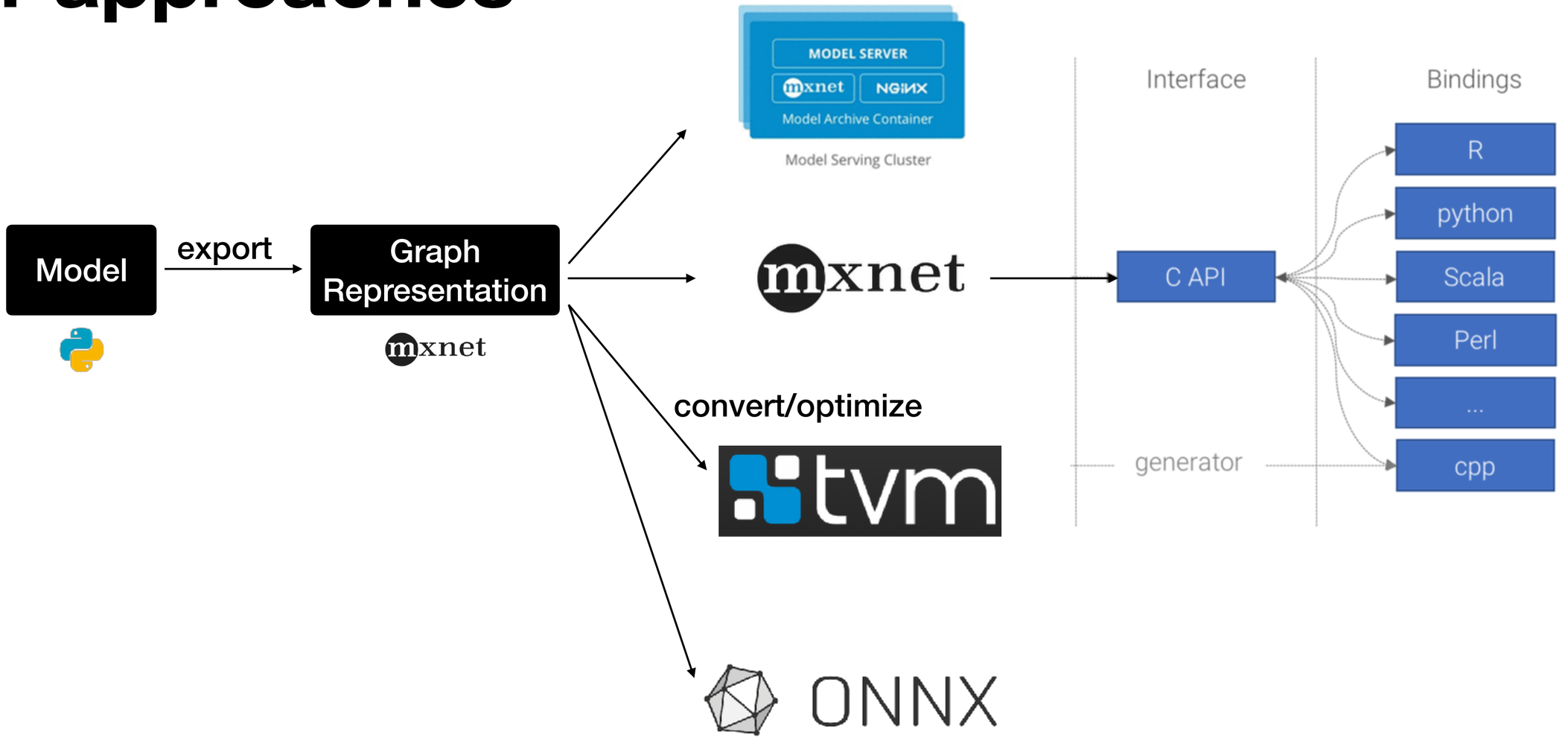
For example, you may train the model on cloud services but deploy the model in a self-driving vehicle with completely different hardware/software stack.



Why deep learning video models are so hard to deploy

- Diverging programming languages in development and deployment
- The gaps of compute power
- Portability and dependency(video)

Our approaches



Jupyter notebook walkthrough

Video Model Inference on Jetson

In [1]:
```
import mxnet as mx
import gluoncv as gcv
import decord
print('mxnet:', mx.__version__)
print('gluoncv:', gcv.__version__)
print('decord:', decord.__version__)
```

```
mxnet: 1.6.0
gluoncv: 0.7.0
decord: 0.3.9
```

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
from mxnet import gluon, nd, image
from mxnet.gluon.data.vision import transforms
from gluoncv.data.transforms import video
from gluoncv import utils
from gluoncv.model_zoo import get_model
```

Preprocessing

In [3]:
```
url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/abseiling_k400.mp4'
video_fname = utils.download(url)
vr = decord.VideoReader(video_fname)
```

Compact example for Jetson on device walkthrough

```
[ 79%] Building CXX object decord/CMakeFiles/decord.dir/src/video/ffmpeg/filter_graph.cc.o
[ 82%] Building CXX object decord/CMakeFiles/decord.dir/src/video/ffmpeg/threaded_decoder.cc.o
[ 86%] Linking CXX shared library libdecord.so
[ 86%] Built target decord
Scanning dependencies of target video_classification
[ 89%] Building CXX object CMakeFiles/video_classification.dir/tvm_runtime_pack.cc.o
[ 93%] Building CXX object CMakeFiles/video_classification.dir/src/classification.cpp.o
[ 96%] Building CXX object CMakeFiles/video_classification.dir/src/video.cpp.o
[100%] Linking CXX executable video_classification
[100%] Built target video_classification
xavier@xavier-a3:~/cvpr20-tutorial/cvpr2020-videomodeling-deployment/tvm_deploy/build$
```

More to expect…

- TVM deployment with 3d Conv networks
- INT8 quantization for GluonCV video models
- GluonCV lite video model zoo: a curated list of models specially optimized for individual devices

Take-aways

- Thinking twice about deployment before you develop the models. Redesign model or rewrite the code for deployment can be costly. We use TVM to generated device specific library which is light weight and efficient.
- We now have `decord` library to deal with video inputs as straightforward as images.
- In case there's CPU/GPU available at the same time, make sure we figure out which one to target. For example, use GPUs instead of CPUs on Jetson devices
- The materials for this tutorial are all open sourced and available at: https://github.com/zhreshold/cvpr2020-videomodeling-deployment

Thank you!