DALI: FAST DATA PIPELINES FOR DEEP LEARNING

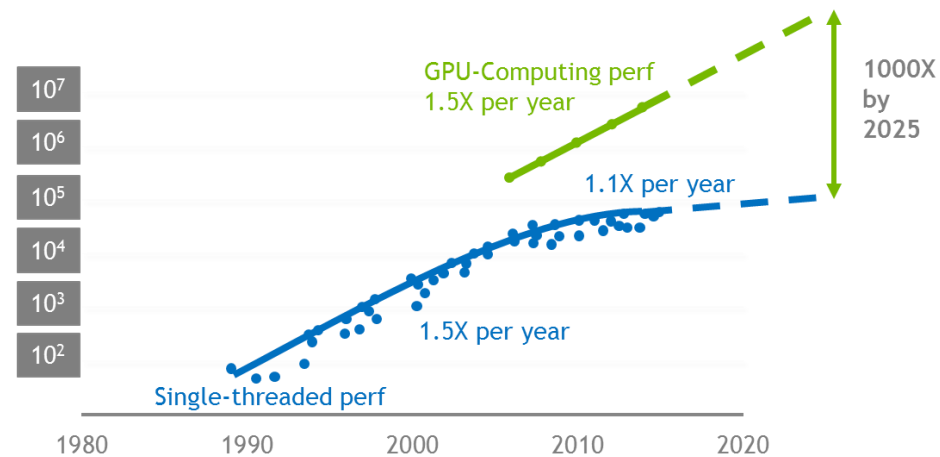
Maitreyi Roy, March 2019

THE PROBLEM

CPU BOTTLENECK OF DL TRAINING

CPU : GPU ratio

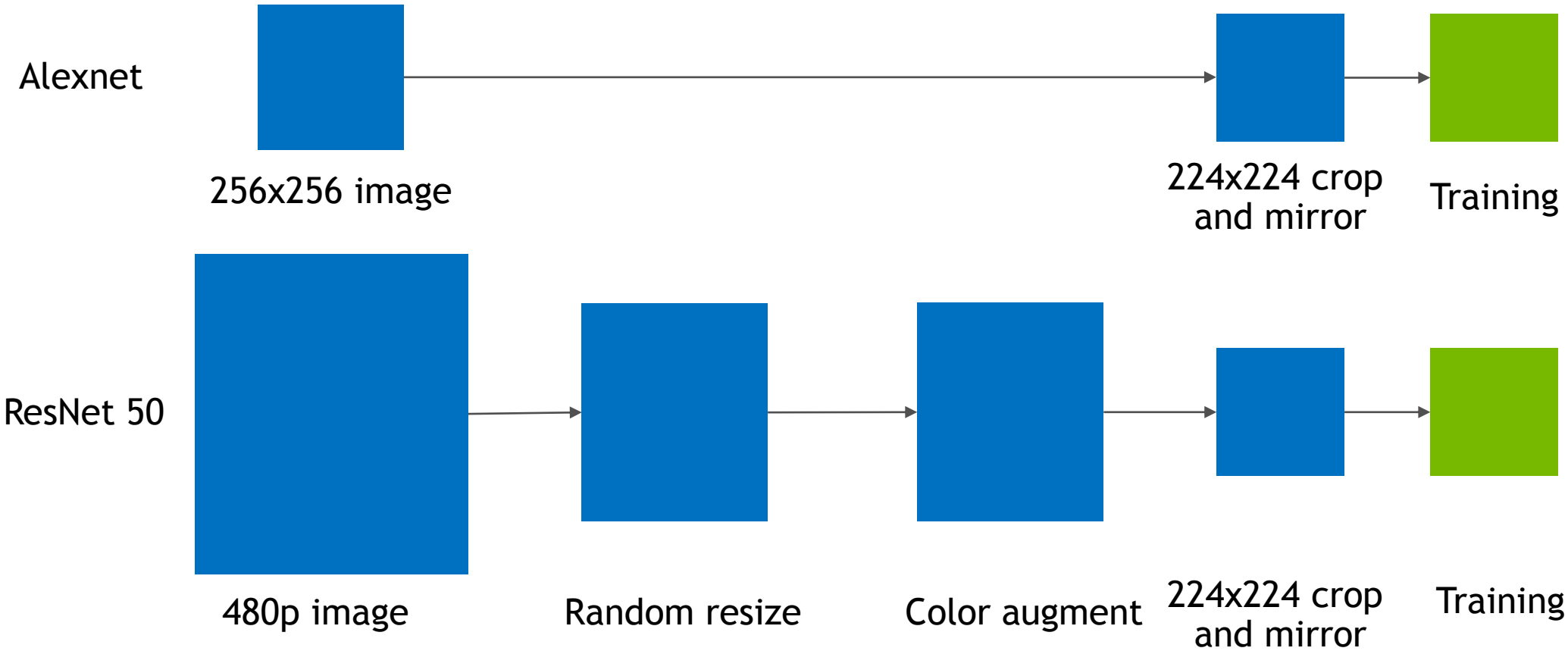
- Multi-GPU, dense systems are more common (DGX-1V, DGX-2)
- Using more cores / sockets is very expensive
- CPU to GPU ratio becomes lower:
  - DGX-1V: **40 cores / 8, 5 cores / GPU**
  - DGX-2: **48 cores / 16, 3 cores / GPU**



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

CPU BOTTLENECK OF DL TRAINING

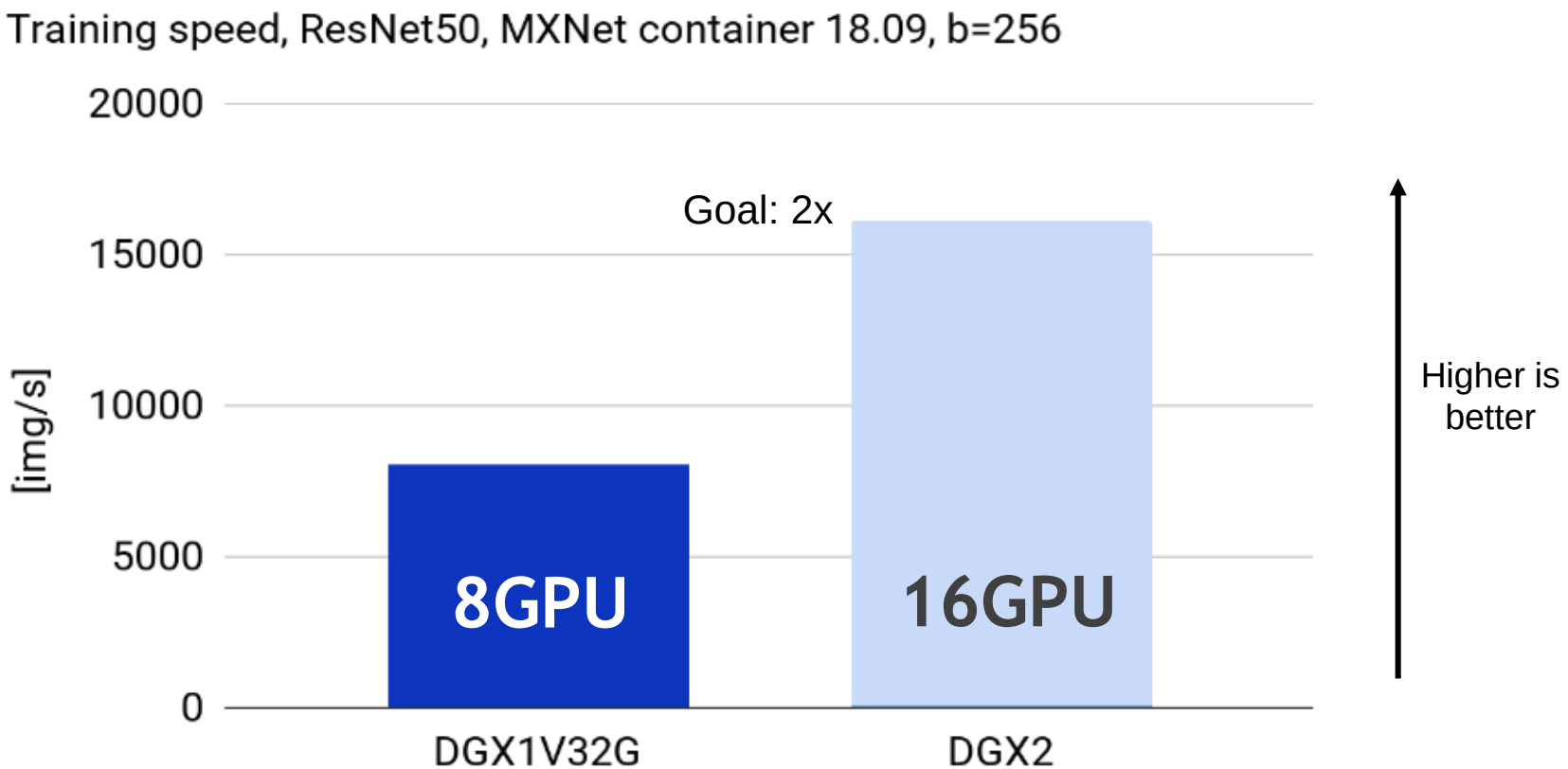
Complexity of I/O pipeline



CPU BOTTLENECK OF DL TRAINING

In practice

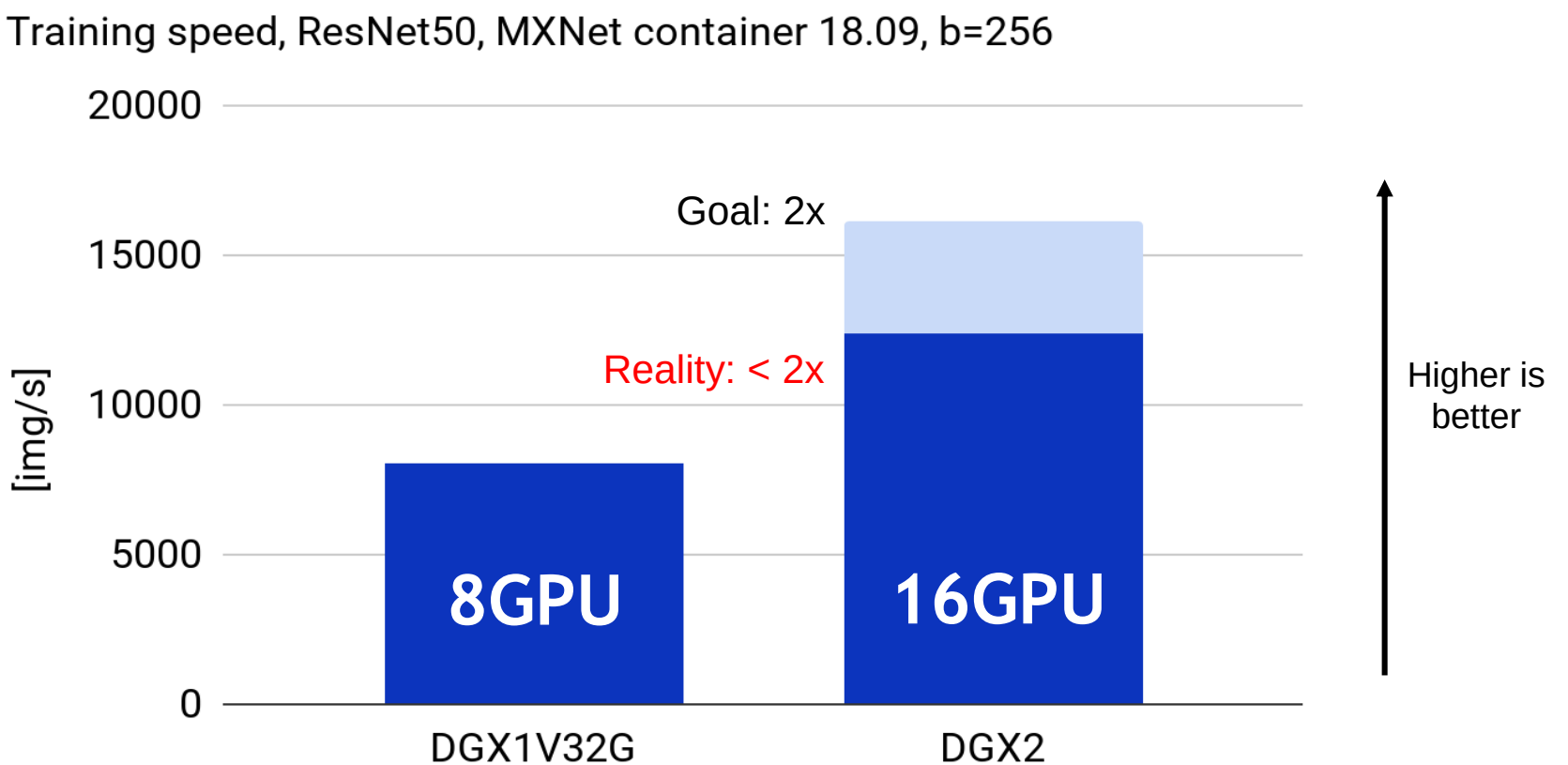
When we put 2x GPU power we don’t get adequate perf improvement



CPU BOTTLENECK OF DL TRAINING

In practice

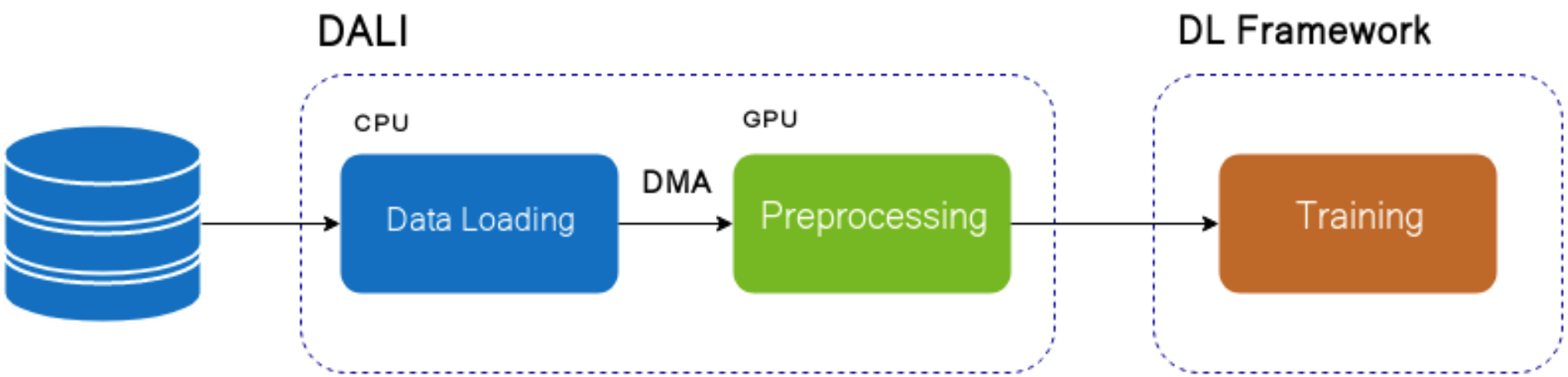
2x GPUs doesn't result in 2x perf improvement



DALI TO THE RESCUE

WHAT IS DALI?

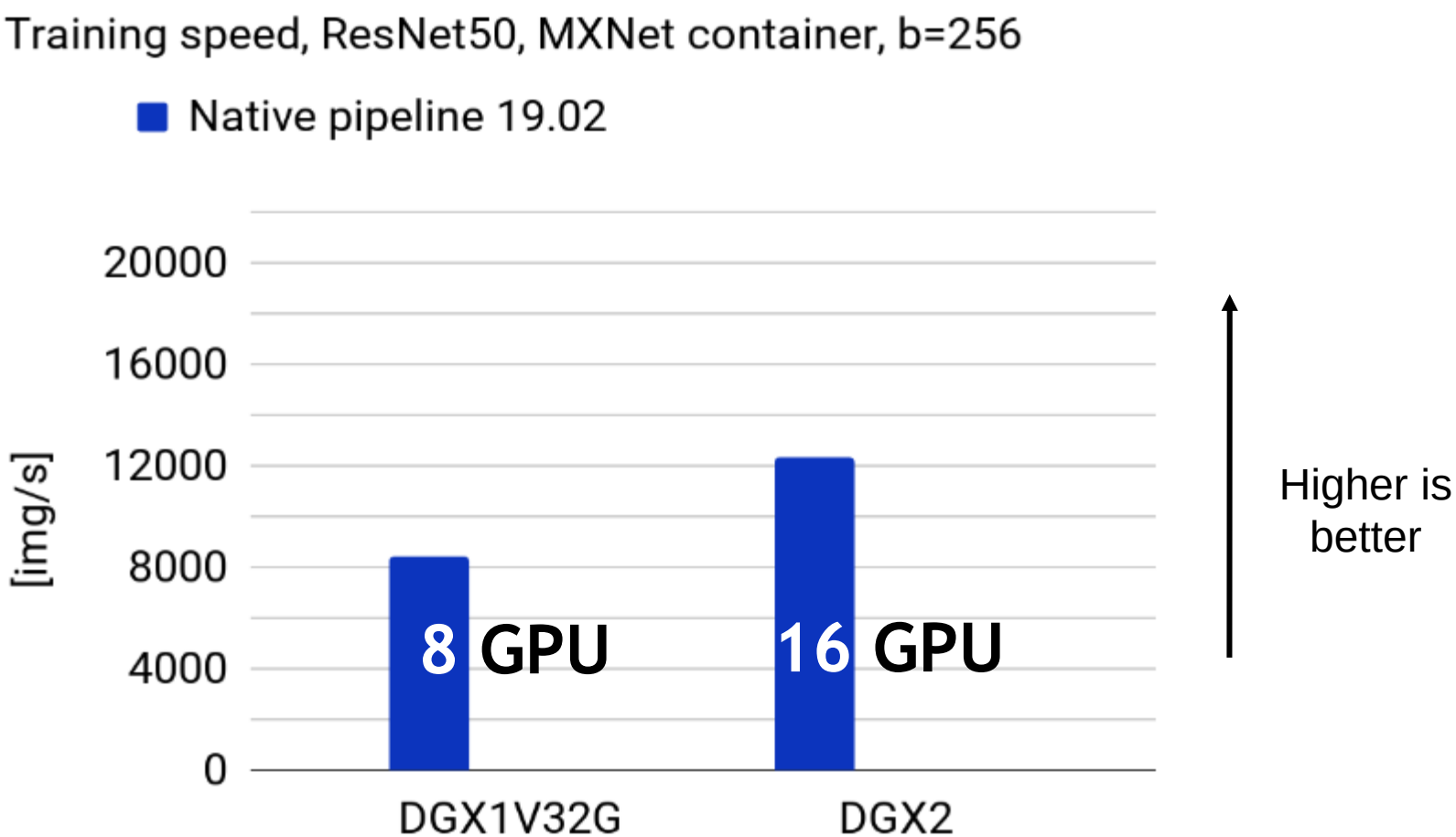
High Performance Data Processing Library



Flexible & Fast

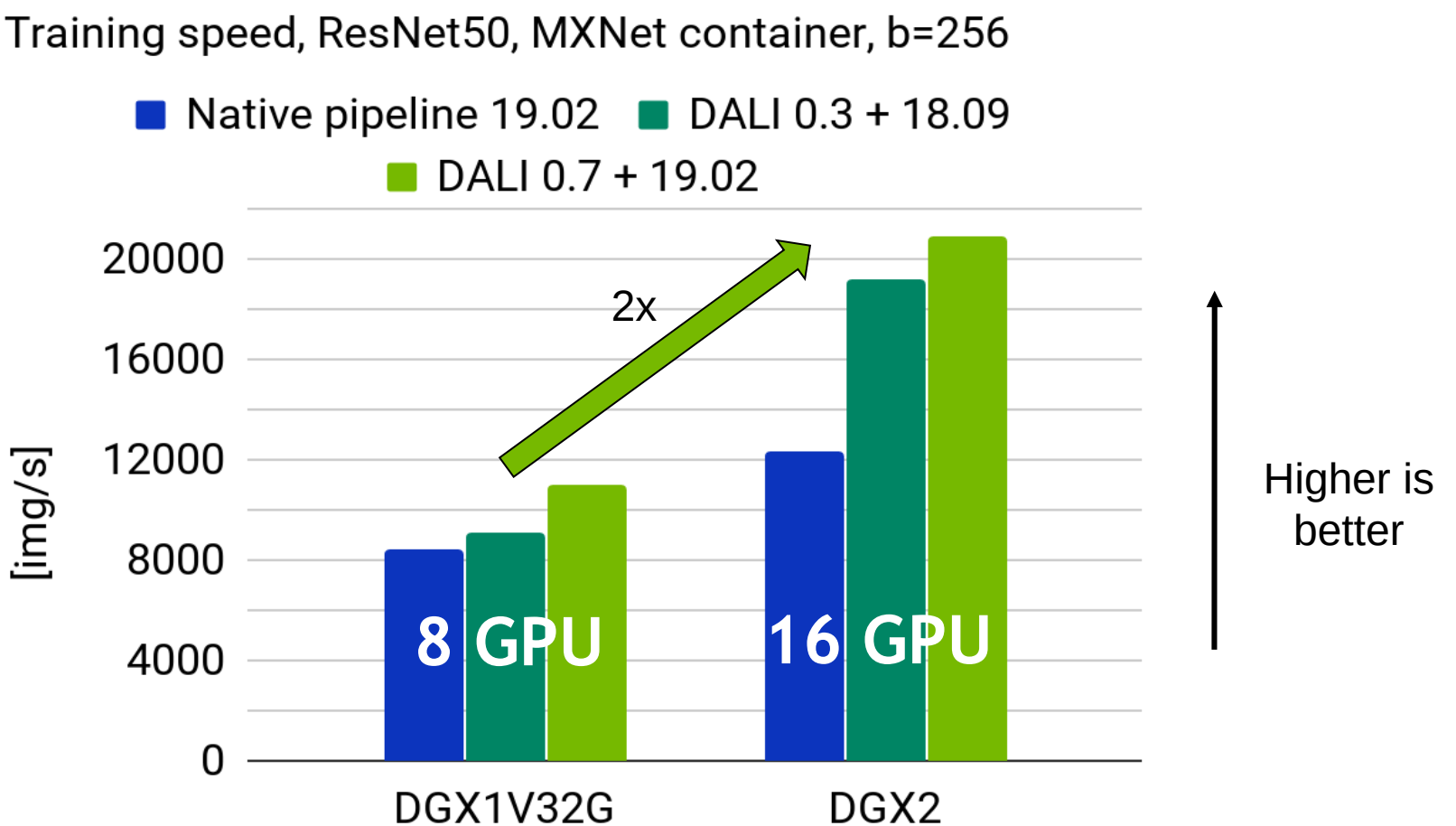
DALI RESULTS

RN50 MXNet



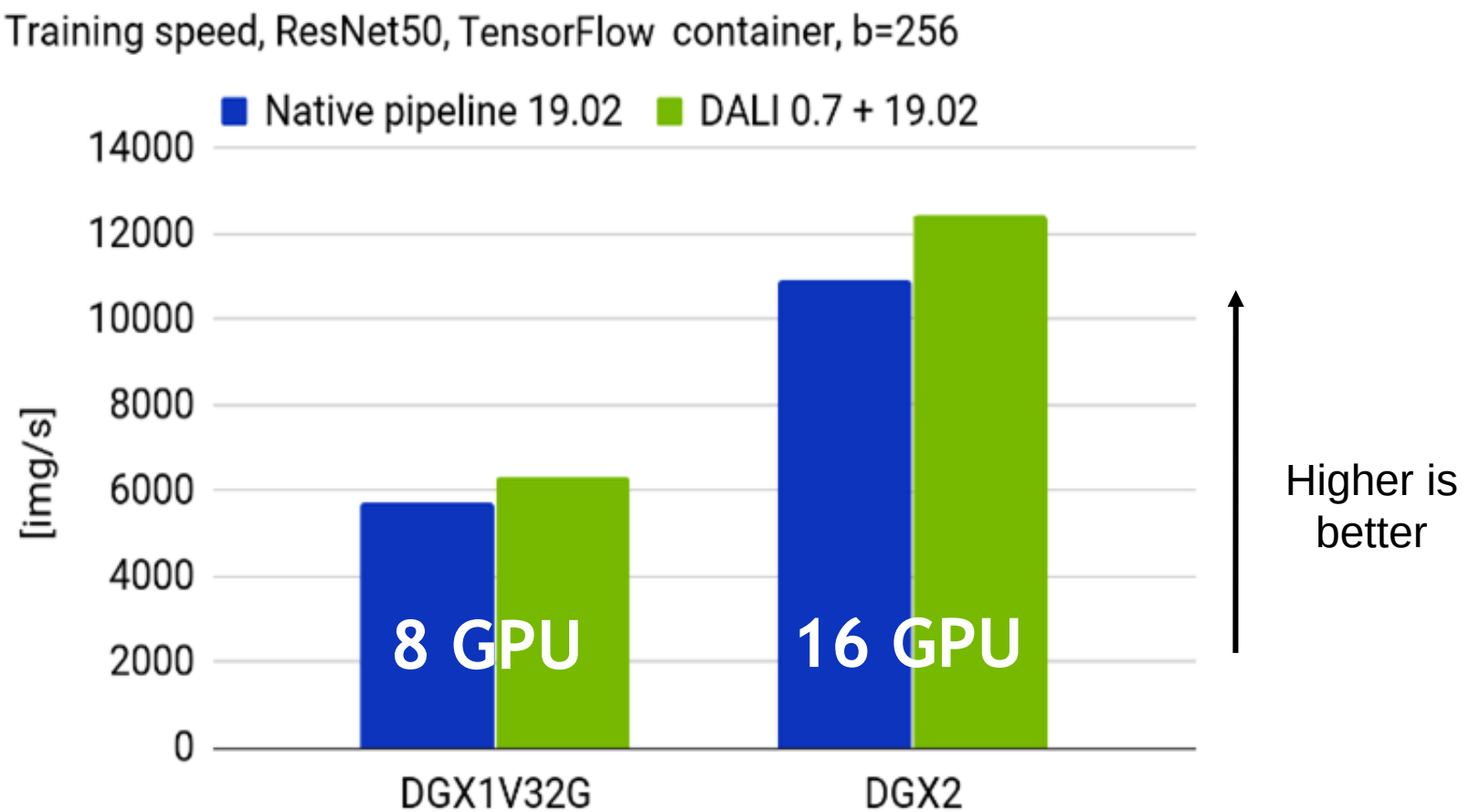
DALI RESULTS

RN50 MXNet



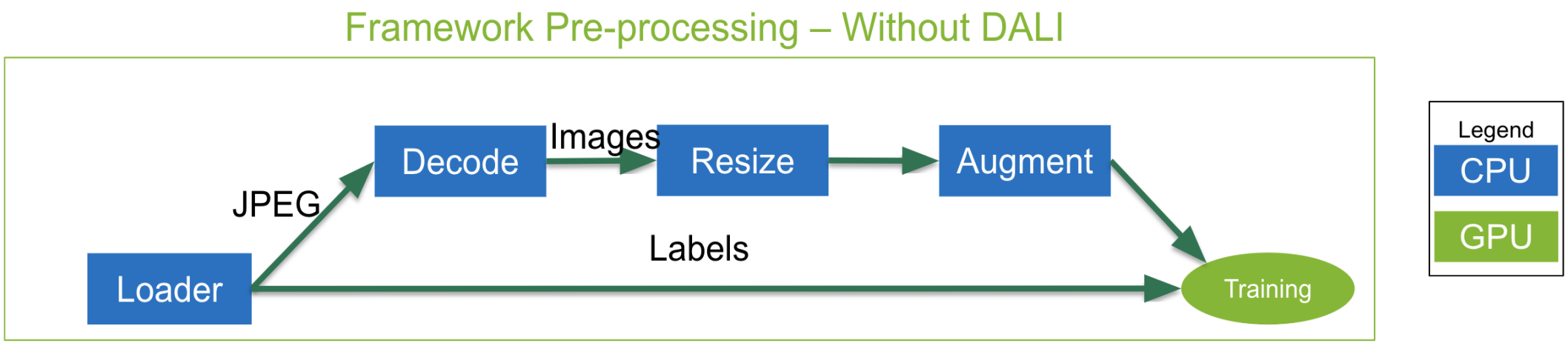
DALI RESULTS

RN50 TensorFlow

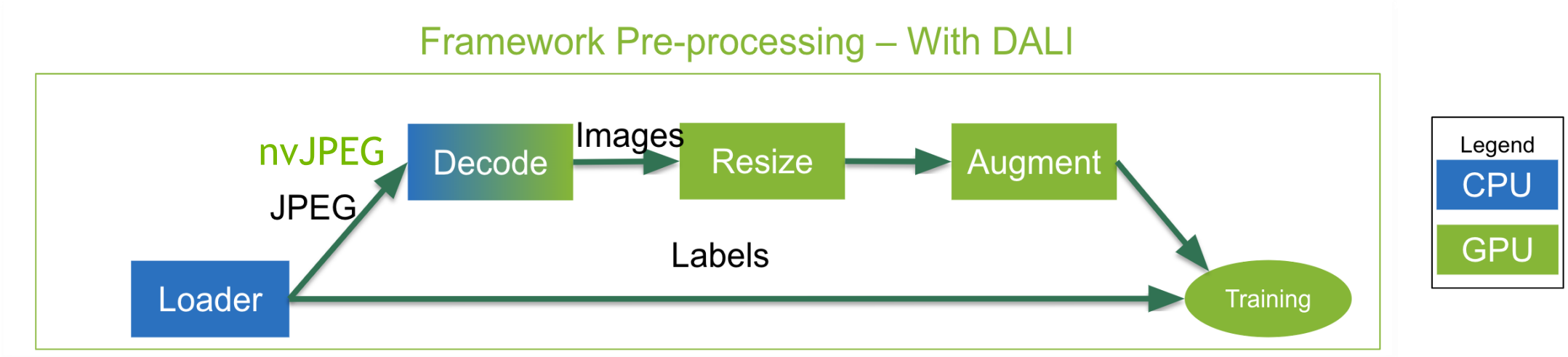


INSIDE DALI

DALI: AN OPTIMIZED GRAPH EXECUTOR



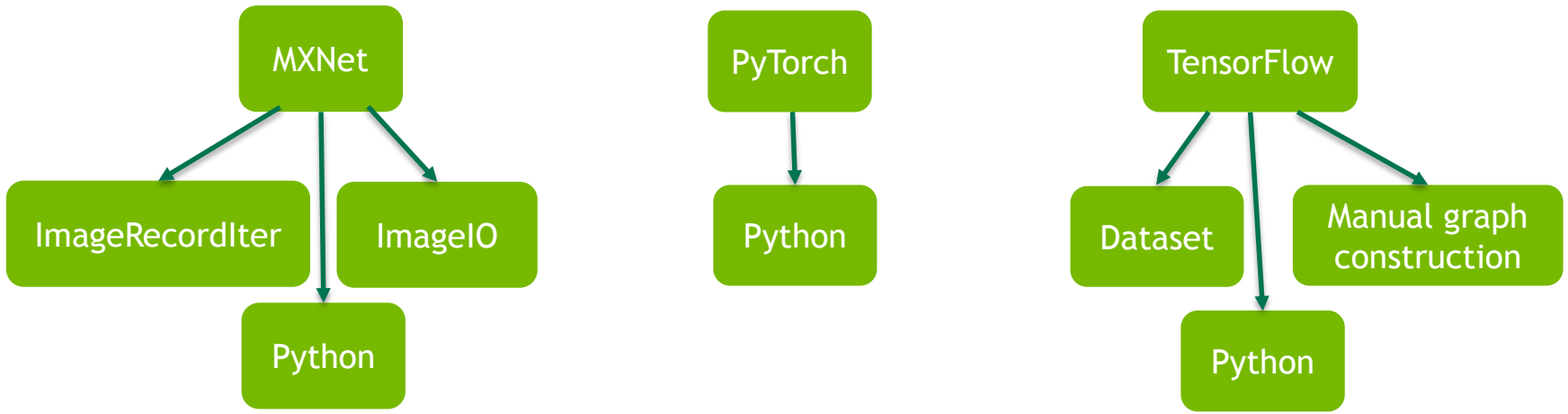
DALI: AN OPTIMIZED GRAPH EXECUTOR



Building and Executing
the graph

LOTS OF FRAMEWORKS

Lots of effort



Frameworks have their own I/O pipelines (often more than 1!)

Tradeoff between performance and flexibility

Lots of duplicated effort to optimize them all

Training process is not portable even if the model is (e.g. via ONNX)

SET YOUR DATA FREE

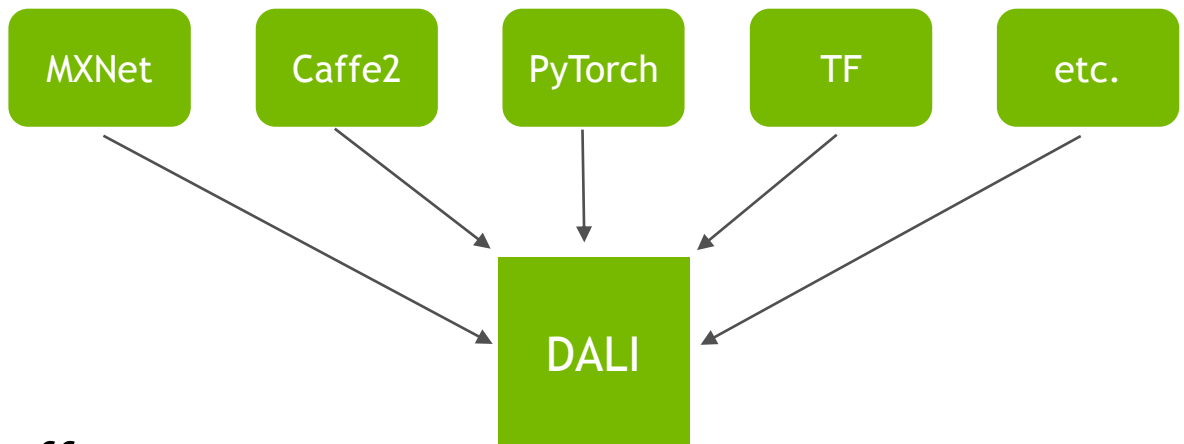
DALI

LMDB (Caffe, Caffe2)	RecordIO (MXNet)	TFRecord (TensorFlow)	List of JPEGs (PyTorch, others)	Video

Use any file format in any framework

SOLUTION: ONE LIBRARY

NVDIA DATA LOADING LIBRARY (DALI)



- Centralize the effort
- Integrate into all frameworks
- Provide both flexibility and performance

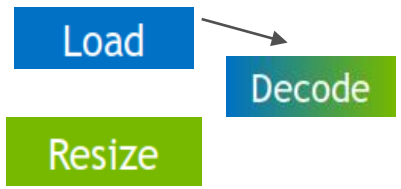
USING DALI

HOW TO USE DALI

Define Graph

Instantiate operators

```
def __init__(self, batch_size, num_threads, device_id):
    super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
    self.input = ops.FileReader(file_root = image_dir)
    self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
    self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
```



Define graph in imperative way

```
def define_graph(self):
    jpegs, labels = self.input()
    images = self.decode(jpegs)
    images = self.resize(images)
    return (images, labels)
```

Use it

```
pipe.build()
images, labels = pipe.run()
```

HOW TO USE DALI

Define Graph

Instantiate operators

```
def __init__(self, batch_size, num_threads, device_id):
    super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
    self.input = ops.FileReader(file_root = image_dir)
    self.decode = ops.nvJPEGDecoder(device = “mixed”, output_type = types.RGB)
    self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
```

Define graph in imperative way

```
def define_graph(self):
    jpegs, labels = self.input()
    images = self.decode(jpegs)
    images = self.resize(images)
    return (images, labels)
```

Use it

```
pipe.build()
images, labels = pipe.run()
```

HOW TO USE DALI

Define Graph

Instantiate operators

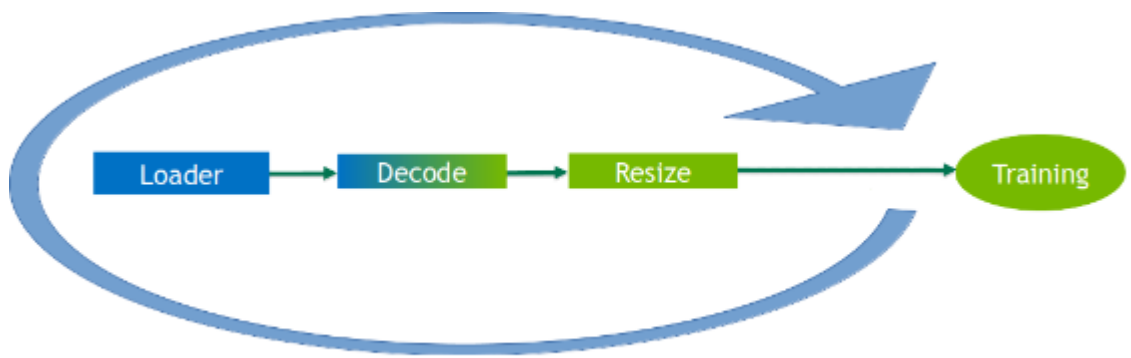
```
def __init__(self, batch_size, num_threads, device_id):
    super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
    self.input = ops.FileReader(file_root = image_dir)
    self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
    self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
```

Define graph in imperative way

```
def define_graph(self):
    jpegs, labels = self.input()
    images = self.decode(jpegs)
    images = self.resize(images)
    return (images, labels)
```

Use it

```
pipe.build()
images, labels = pipe.run()
```



HOW TO USE DALI

Define Graph

Instantiate operators

```
def __init__(self, batch_size, num_threads, device_id):
    super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
    self.input = ops.FileReader(file_root = image_dir)
    self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
    self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
```

Define graph in imperative way

```
def define_graph(self):
    jpegs, labels = self.input()
    images = self.decode(jpegs)
    images = self.resize(images)
    return (images, labels)
```

Use it

```
pipe.build()
images, labels = pipe.run()
```

HOW TO USE DALI

Define the Pipeline

```
class SimplePipeline(nvidia.dali.Pipeline):

    def __init__(self, batch_size, num_threads, device_id):

        super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
        self.input = ops.FileReader(file_root = image_dir)
        self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
        self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
    def define_graph(self):

        jpegs, labels = self.input()
        images = self.decode(jpegs)
        images = self.resize(images)
        return (images, labels)
```

HOW TO USE DALI

Define the Pipeline

```
class SimplePipeline(nvidia.dali.Pipeline):

    def __init__(self, batch_size, num_threads, device_id):

        super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
        self.input = ops.FileReader(file_root = image_dir)
        self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
        self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
    def define_graph(self):

        jpegs, labels = self.input()
        images = self.decode(jpegs)
        images = self.resize(images)
        return (images, labels)
```

HOW TO USE DALI

Define the Pipeline

```
class SimplePipeline(nvidia.dali.Pipeline):

    def __init__(self, batch_size, num_threads, device_id):

        super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
        self.input = ops.MxNetReader(index_path = idx, path = path)
        self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
        self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
    def define_graph(self):

        jpegs, labels = self.input()
        images = self.decode(jpegs)
        images = self.resize(images)
        return (images, labels)
```

HOW TO USE DALI

Define the Pipeline

```
class SimplePipeline(nvidia.dali.Pipeline):

    def __init__(self, batch_size, num_threads, device_id):

        super(SimplePipeline, self).__init__(batch_size, num_threads, device_id)
        self.input = ops.TFRecordReader(index_path = idx, path = path)
        self.decode = ops.nvJPEGDecoder(device = "mixed", output_type = types.RGB)
        self.resize = ops.Resize(device = "gpu", resize_x = 224, resize_y = 224)
    def define_graph(self):

        jpegs, labels = self.input()
        images = self.decode(jpegs)
        images = self.resize(images)
        return (images, labels)
```

HOW TO USE DALI

Use in PyTorch: Few Lines of Code

PyTorch DataLoader

```
train_loader = torch.utils.data.DataLoader(...)
prefetcher = data_prefetcher(train_loader)
input, target = prefetcher.next()
i = -1
while input is not None:
    i += 1
    (...)
    input, target = prefetcher.next()
```

DALI iterator

```
dali_pipe = TrainPipe(...)
train_loader = DALIClassificationIterator(dali_pipe)


for i, data in enumerate(train_loader):
    input = data[0]["data"]
    target = data[0]["label"].squeeze()
    (...)
```

HOW TO USE DALI

Use in MXNet: Few Lines of Code

MXNet DataIter and DataBatch

```
train_data = SyntheticDataIter(...)


for i, batches in enumerate(train_data):
   data = [b.data[0] for b in batches]
   label =
[b.label[0].as_in_context(b.data[0].context) for b
in batches]
   (...)
```

DALI iterator

```
dali_pipes = [TrainPipe(...) for gpu_id in gpus]
train_data = DALIClassificationIterator(dali_pipe)

for i, batches in enumerate(train_data):
   data = [b.data[0] for b in batches]
   label =
[b.label[0].as_in_context(b.data[0].context) for b in
batches]
   (...)
```

HOW TO USE DALI

Use in TensorFlow

TensorFlow Dataset

```
def get_data():
    ds = tf.data.Dataset.from_tensor_slices(files)
    ds.define_operations(...)
    return ds




classifier.train(input_fn=get_data,...)
```

DALI TensorFlow operator

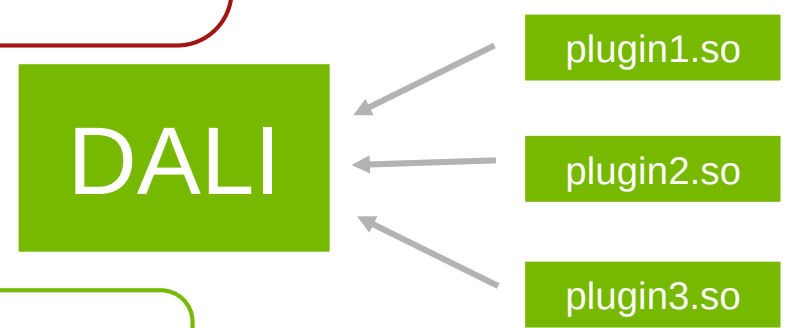
```
def get_data():
    dali_pipe = TrainPipe(...)
    daliop = dali_tf.DALIIterator()
    with tf.device("/gpu:0"):
        img, labels = daliop(pipeline=dali_pipe, ...)
    return img, labels

classifier.train(input_fn=get_data,...)
```

PLUGIN MANAGER

Adds Extensibility

Create operator

```
template<>
void Dummy<GPUBackend>::RunImpl(DeviceWorkspace *ws, const int idx) {
(...)
}
DALI_REGISTER_OPERATOR(CustomDummy, Dummy<GPUBackend>, GPU);
```



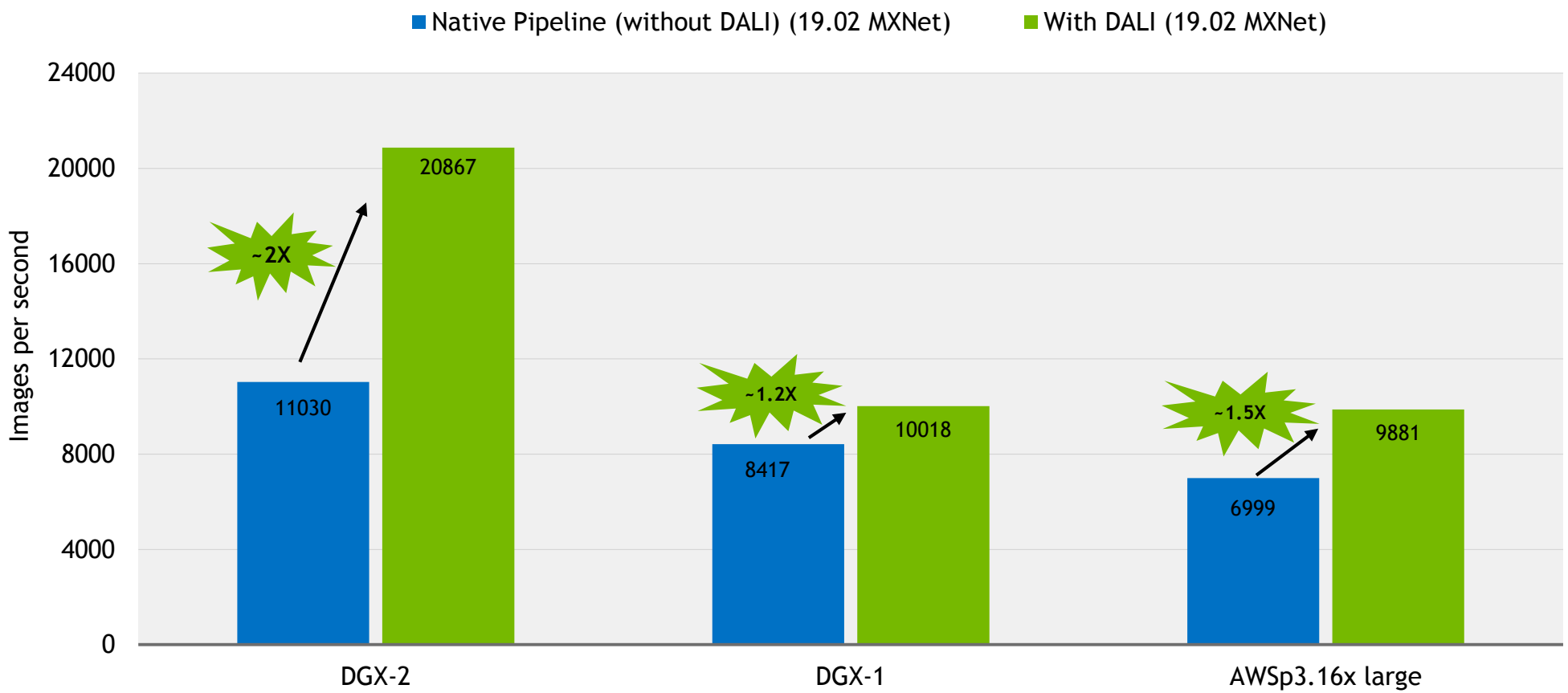
Load Plugin from python

```
import nvidia.dali.plugin_manager as plugin_manager
plugin_manager.load_library('./customdummy/build/libcustomdummy.so')
ops.CustomDummy(...)
```

USE CASES

DALI 0.7 PERFORMANCE

Training ResNet50 on MXNet



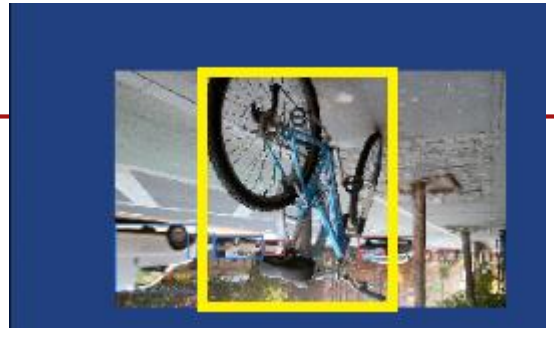
MXNet 19.02 NGC Container, batch size: 256 for DGX-2 and DGX-1, batch size: 192 for AWSp3.16x large

OBJECT DETECTION

Single Shot Multibox Detector Model (SSD)

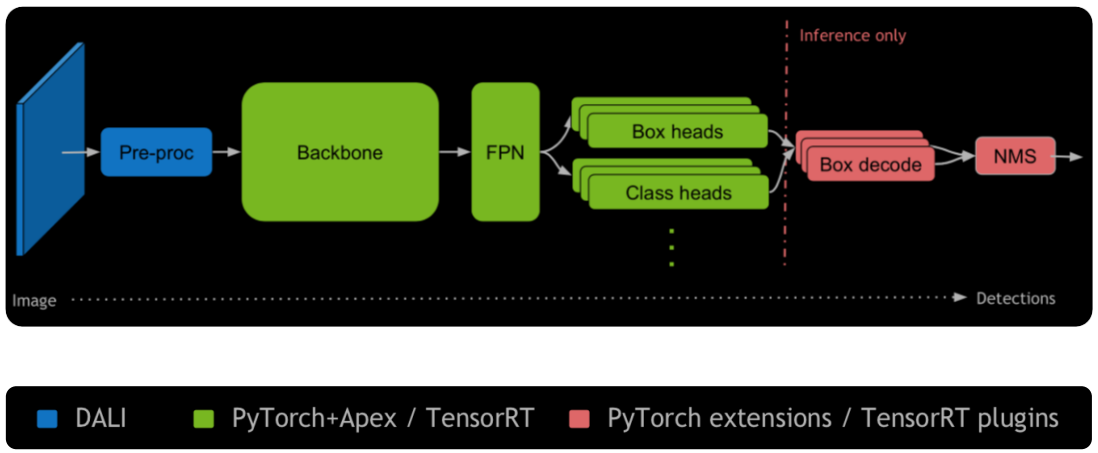
Use operators in the DALI graph:

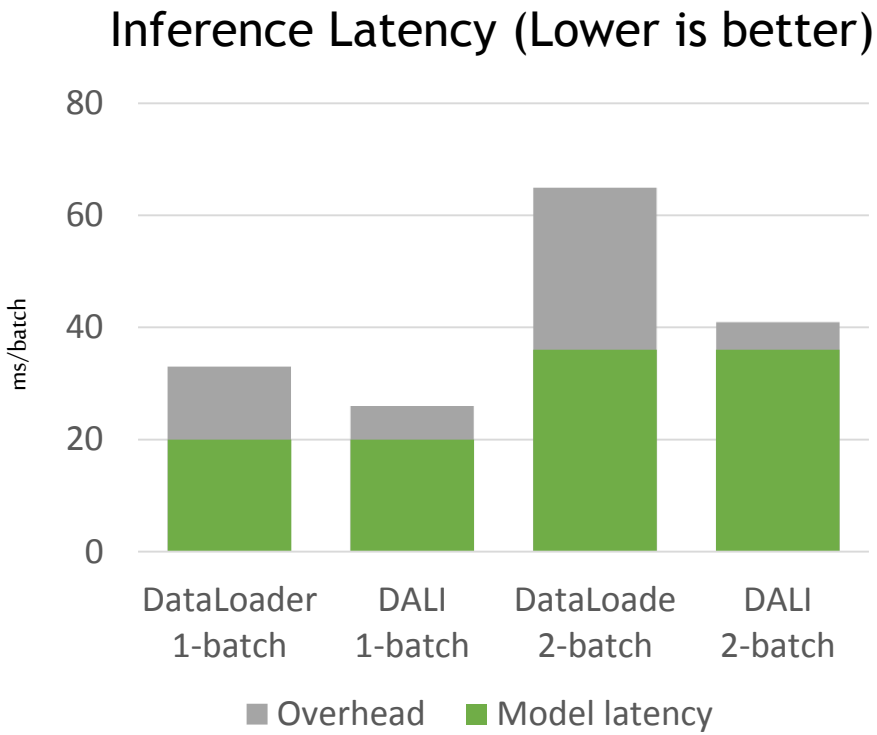
```
images = self.paste(images, paste_x = px, paste_y = py, ratio = ratio)
bboxes = self.bbpaste(bboxes, paste_x = px, paste_y = py, ratio = ratio)

crop_begin, crop_size, bboxes, labels = self.prospective_crop(bboxes, labels)
images = self.slice(images, crop_begin, crop_size)

images = self.flip(images, horizontal = rng, vertical = rng2)
bboxes = self.bbflip(bboxes, horizontal = rng, vertical = rng2)

return (images, bboxes, labels)
```



OBJECT DETECTION

RetitnaNet ONNX-TensorRT Training / Inference





Creating Object Detection Pipeline for GPUs
https://devblogs.nvidia.com/object-detection-pipeline-gpus/
s9243-fast-and-accurate-object-detection-with-pytorch-and-tensorrt

VIDEO

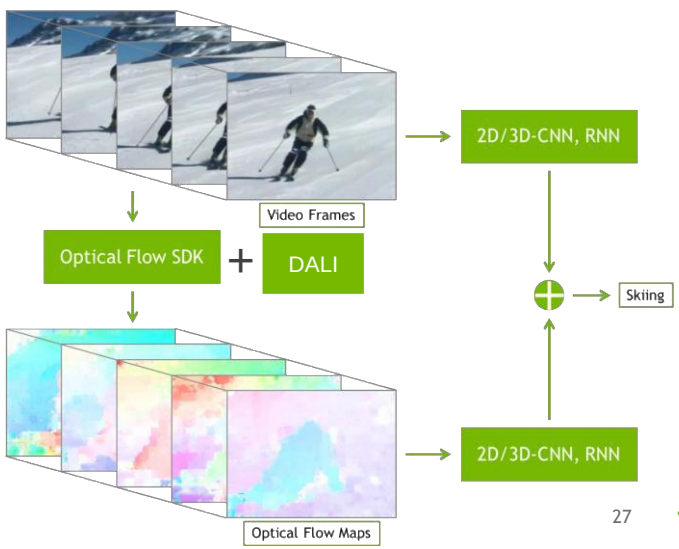
Optical Flow Example

Instantiate operator:

```
self.input = ops.VideoReader(file_root = video_files, sequence_length = len, step = step)
self.opticalFlow = ops.OpticalFlow()
self.takeFirst = ops.ElementExtract(element_map = [0])
```

Use it in the DALI graph:

```
frames = self.input()
flow = self.opticalFlow(frames)
first = self.takeFirst(frames)
return first, flow
```



MAKING LIFE EASIER

NVIDIA DATA LOADING LIBRARY (DALI)

Fast Data Processing Library for Accelerating Deep Learning

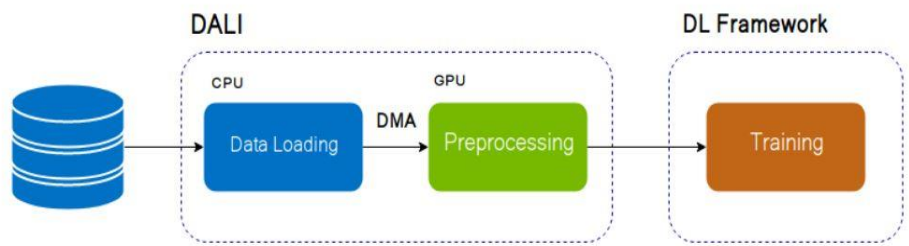
Full input pipeline acceleration including data loading and augmentation

Drop-in integration with DL frameworks

Portable workflows through multiple input formats and configurable graphs

Flexible through configurable graphs and custom operators

DALI in DL Training Workflow



Currently supports:

- ResNet50 (Image Classification), SSD (Object Detection)
- Input Formats – JPEG, LMDB, RecordIO, TFRecord, COCO, H.264, HVEC
- Python/C++ APIs to define, build & run an input pipeline

Over **1100** GitHub stars | Top 50 ML Projects (out of 22,000 in 2018)

MORE EXAMPLES

Help you get started

ResNet50 for PyTorch mxnet TensorFlow

How to read data in various frameworks

How to create custom operators

Pipeline for the detection

Video pipeline

And more to come...

Documentation and tutorials available online:

https://docs.nvidia.com/deeplearning/sdk/dali-developer-guide/docs/index.html

DALI

Summary

- Open source, GPU-accelerated data augmentation and image loading library
  DALI focuses on performance, flexibility and training portability
- DL customers training on 8x GPUs, are likely bottlenecked on CPUs
- Download and evaluate DALI (NGC containers, pip whl, open source)
- Full pre-processing data pipeline ready for training and inference
- Easy framework integration
- Portable training workflows

FUTURE

- Full Video Support (video classification, detection, slo-mo, upscale, denoise, super-res)
- Extract augmentation operators in a separate library (inference,…)
- 3D volumetric data
- More workloads (segmentation)

DALI RESOURCES

- Official Documentation (Quick Start, Developer Guides)
  - https://docs.nvidia.com/deeplearning/sdk/index.html#data-loading
- GitHub Documentation
  - https://docs.nvidia.com/deeplearning/sdk/dali-developer-guide/docs/index.html
- DALI Samples & Tutorial
  - https://docs.nvidia.com/deeplearning/sdk/dali-developer-guide/docs/examples/index.html
- DALI Blog
  - https://devblogs.nvidia.com/fast-ai-data-preprocessing-with-nvidia-dali/
- DALI talk at GTC 2019
  - https://on-demand-gtc.gputechconf.com/gtcnew/sessionview.php?sessionName=s9925-fast+ai+data+pre-processing+with+nvidia+dali


NVIDIA