Improving Caffe: Some Refactoring

Yangqing Jia

Dependencies?

leveldb

lmdb

OpenCV

hdf5

Boost

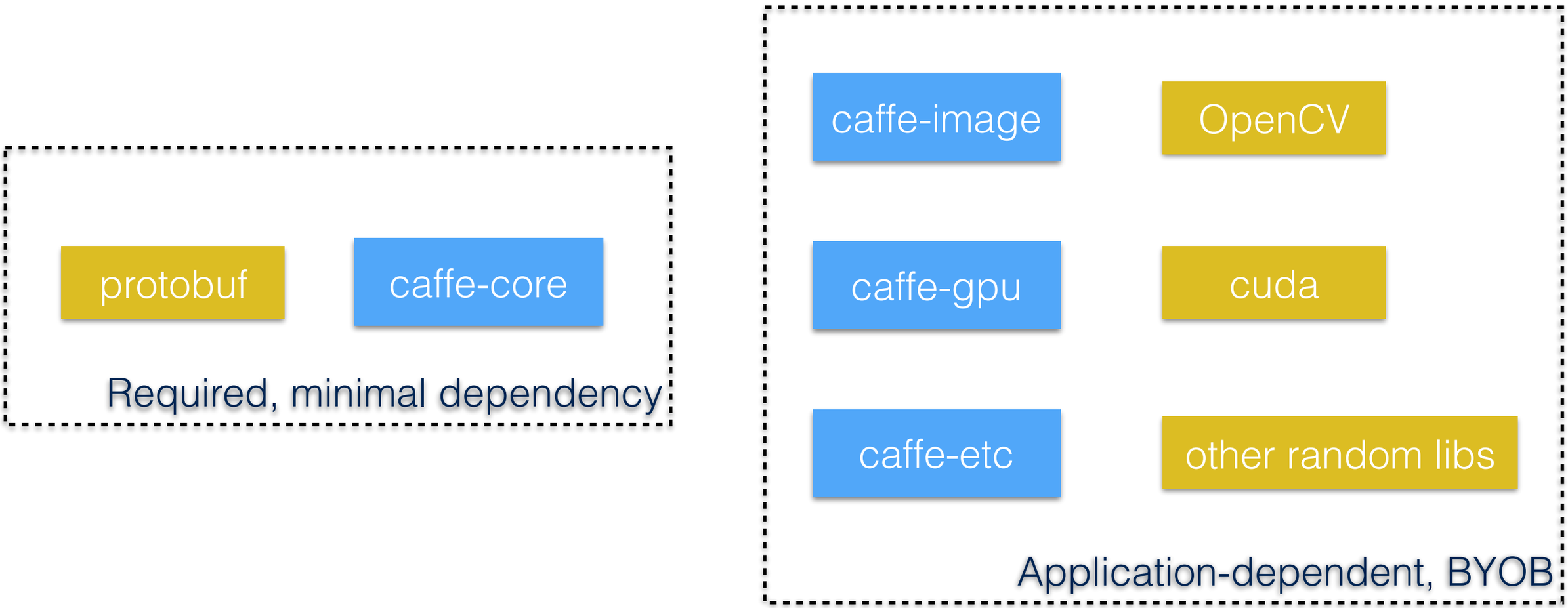
cuda

gflags

protobuf

Caffe

Dependency!


caffe-image
OpenCV
protobuf
caffe-core
caffe-gpu
cuda
caffe-etc
other random libs
Required, minimal dependency
Application-dependent, BYOB

Devices

- Current Caffe

```
class Layer {
    void Forward_cpu();
    void Forward_gpu();
    void Backward_cpu();
    void Backward_gpu();
}
```

- Simplified Interface

```
template <typename Device>
class Layer {
    void Forward(); void Backward()
}
```

Blob is Anything

- Current Caffe

```
template <typename Dtype>
class Blob {
    Dtype* data; …
};
```

- Simplified Version

```
class Blob {
    AnyPointer data;
    DataType type;
};
```

- Keep the relaxed Blob is N-dimensional generality.
  (N-D Blobs in Caffe master)

Operators

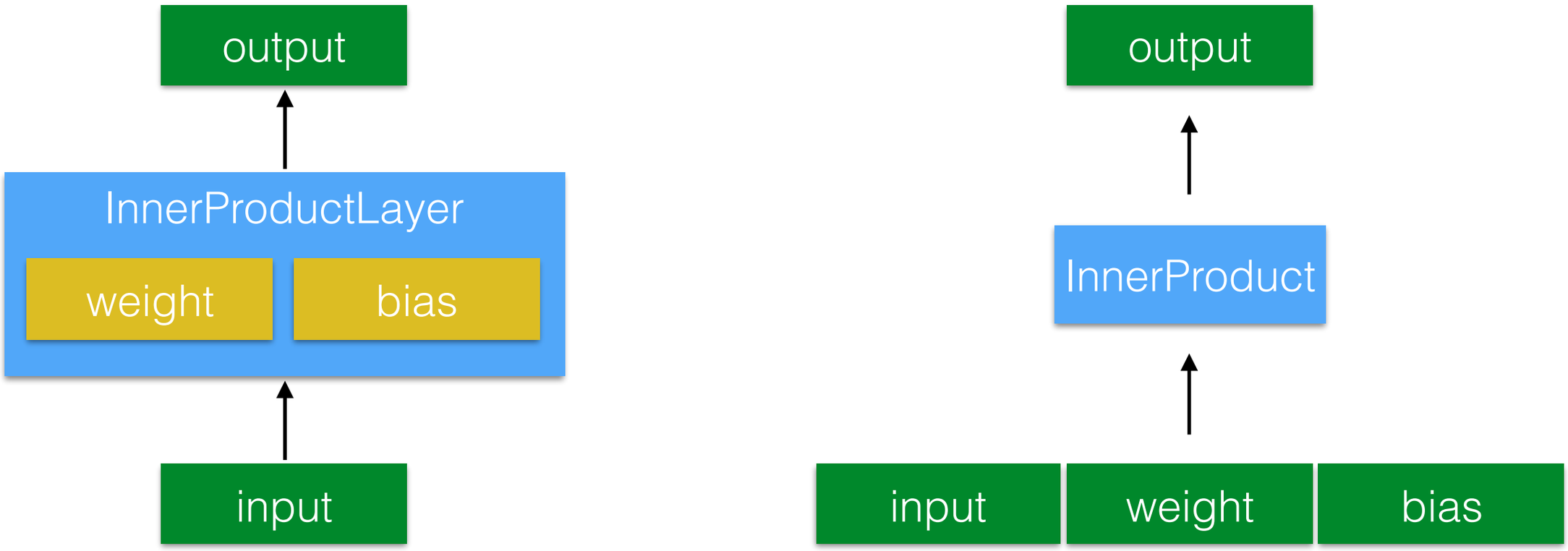
- Current Caffe

```
class Layer {
    void Forward();
    void Backward();
};
```

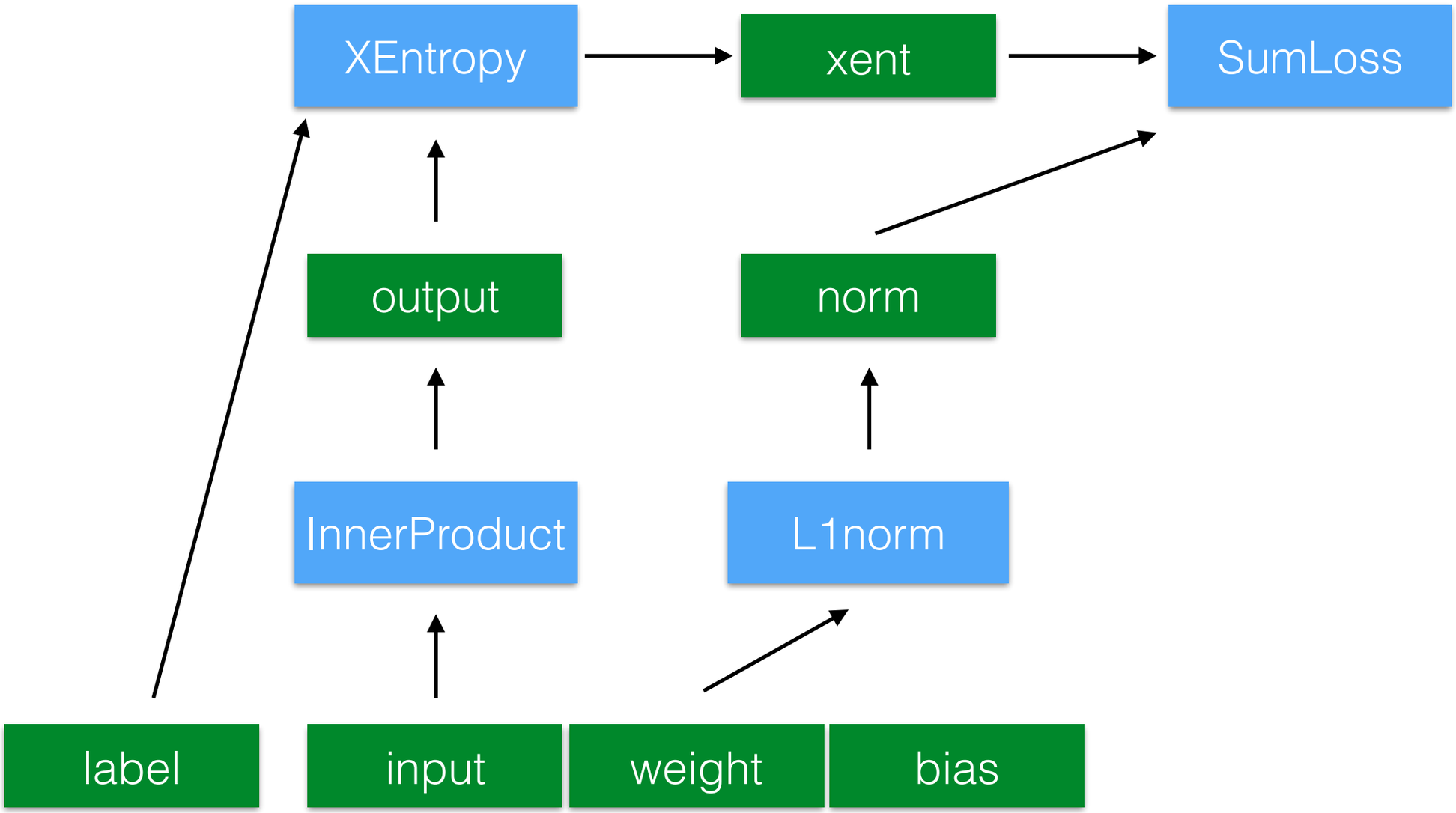
- Simplified Interface

```
class Operator {
    void Run();
};
```

Which means…


output
InnerProductLayer
weight
bias
input
output
InnerProduct
input
weight
bias

Easier Algorithms



Explicit Gradients

- Current Caffe

```
class Layer {
    void Forward();
    void Backward();
};
```

- I’m playing with…

```
@GradientRegistry.RegisterGradient("Relu")
def ReluGradient(op):
    return CreateOperator("ReluGradient")
```

Solvers are Ops Too

```
for param in [filter1, bias1, filter2, bias2]:
  net.WeightedSum([param, ONE, param.Grad(), LR], param)
```

Ops for SGD

```
# Add weight decay.
for param in [filter1, filter2]:
    net.WeightedSum([param.Grad(), ONE, param, WEIGHT_DECAY],
                    param.Grad())
# do momentum
for param in [filter1, bias1, filter2, bias2]:
    net.Accumulate([param.Grad()], param.Momentum(), gamma=0.9)
# update
for param in [filter1, bias1, filter2, bias2]:
    net.WeightedSum([param, ONE, param.Momentum(), LR], param)
```

Use Cases

- Accumulate gradients
  multiple forward and backward + single update

- Multi-GPU?
  computation with communication on the side

- Quick research runs
  a custom solver, etc.

Purpose…

- Faster Experimenting and Prototyping
- Simpler Compilation
- More Portability