

Tensorfolding: Improving Performance of Convolutional Neural Networks with Fused Microkernels

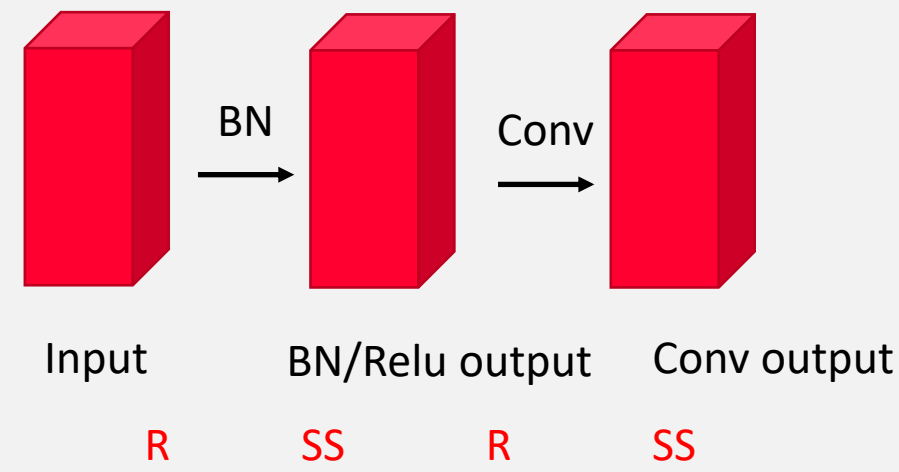
Michael Anderson, Evangelos Georganas , Sasikanth Avancha , and Alexander Heinecke,
Parallel Computing Lab, Intel Labs
{michael.j.anderson, evangelos.georganas, sasikanth.avancha , alexander.heinecke}@intel.com

Overview

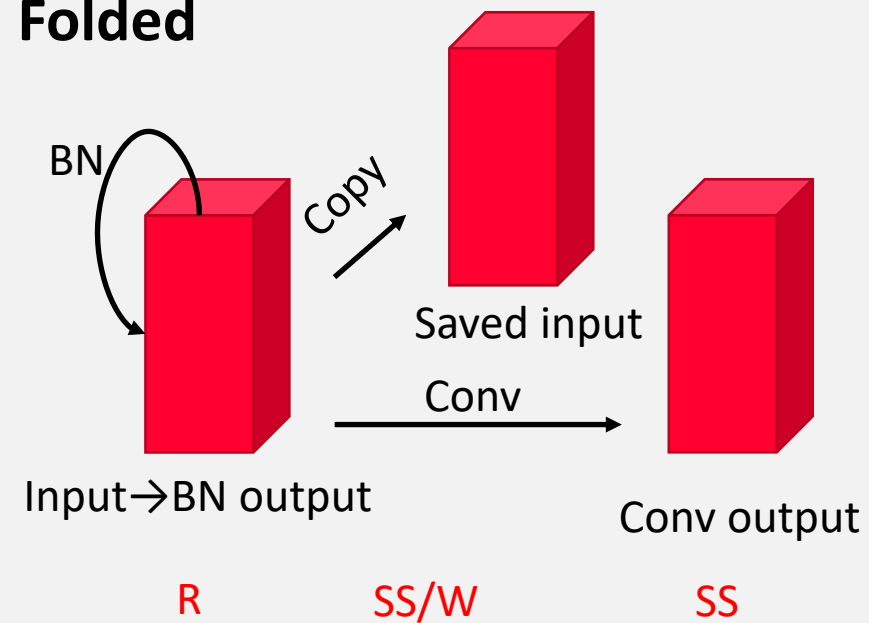
- Convolutional Neural Networks include
 - computationally-intensive convolutional layers (Conv) and
 - bandwidth-bound non-linear elementwise operations, such as batch normalization (BN), ReLU, etc.
- In many cases, layer fusion allows for avoiding unnecessary reads and writes to memory [1]
- However, there are limits to layer fusion, as some tensors are needed in the backward pass and must be written and/or reduction operations are involved (BN)
- We explore techniques for hiding these reads and writes behind convolutional layers to exploit better computation/communication overlap

Example: Conv+BN+ReLU

Fused



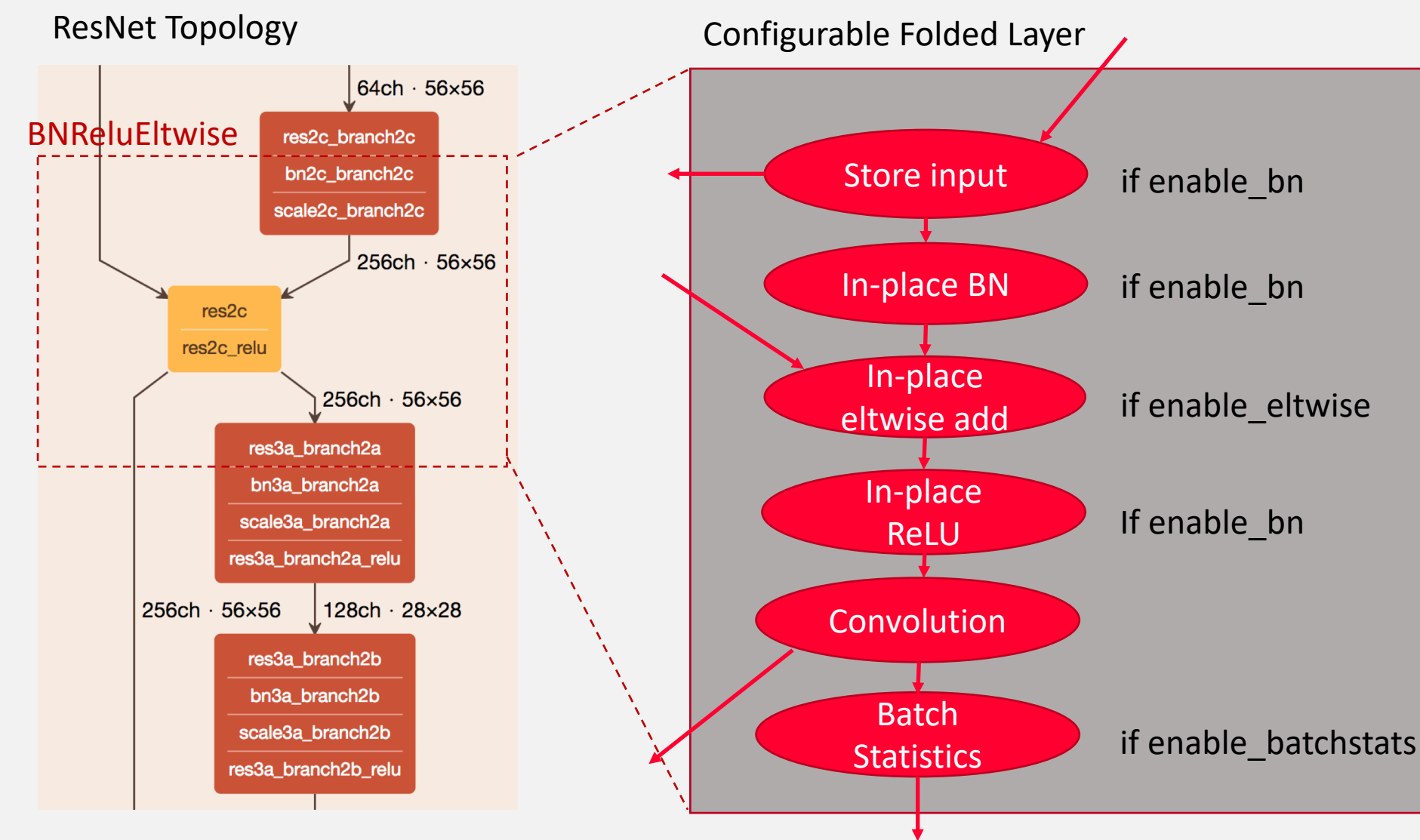
Folded



- Both cases incur the same amount of memory traffic (4 streams)
- Folded version allows memory traffic to overlap with compute, a latency optimization essential on CPUs

Layer Implementation

- For forward pass, we construct a configurable and folded layer including BN, elementwise, convolution, and batch statistics
- These features are enabled or disabled with runtime flags and potentially handled with JIT-ed code
- Three configurations used for ResNet:
 - NoBNRelu:** $(enable_bn, enable_eltwise, enable_batchstats) = (0,0,1)$
 - BNRelu:** $(enable_bn, enable_eltwise, enable_batchstats) = (1,0,1)$
 - BNReluEltwise:** $(enable_bn, enable_eltwise, enable_batchstats) = (1,1,1)$



Kernel Streams

- LIBXSMM [2] Includes a deep learning layer generator, based on JIT compiled small GEMM kernels
- Layers are specified by a stream of microkernels (e.g. convolution) which are interpreted and executed efficiently in a linear fashion
 - Example microkernel: apply batch norm to a specific range of pixels
- With this technique, we can interleave (fold) the non-convolution operations at different levels of the loop nest
- We also create custom JIT compiled convolutions that fuse the above operations while leaving data in registers

Dryrun pseudocode

```
for (img=0...
  for (ifmb= 0...
    ...
    for(oj=0...
      for(oi=0...
        if(ofm==0) // IFM first touch
          stream.append(BatchNormRelu(...))
          stream.append(Conv1x1(...))
```

Generated Kernel Stream

```
<BatchNormRelu oi_start=0 oj_start=0 oi_end=56 oj_end=1.../>
<Conv1x1 oi_start=0 oj_start=0 oi_end=28 oj_end=1.../>
<Conv1x1 oi_start=28 oj_start=0 oi_end=56 oj_end=1.../>
<BatchNormRelu oi_start=0 oj_start=1 oi_end=56 oj_end=2.../>
...
```

Experiment

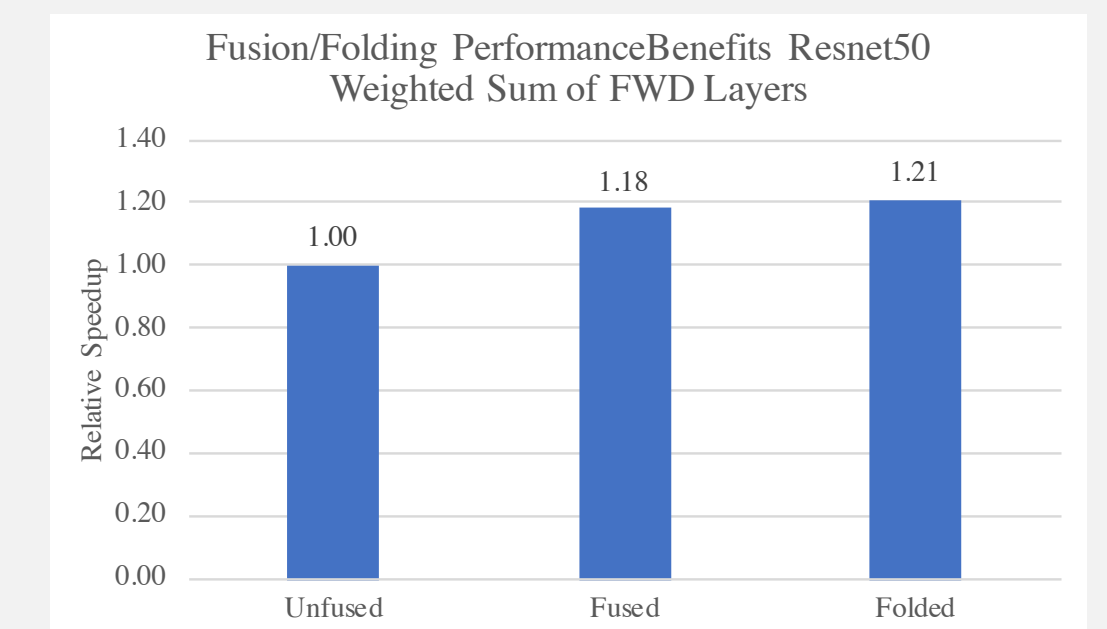
- We compare four levels of folding:
 - Unfused:* Batch norm done in a two passes, batch norm and ReLU done in two passes before convolution
 - Fused:* Single-pass batch norm + ReLU, convolution, then fused batch statistics
 - Folding:* Batch norm the inputs to the convolution, stream store the inputs, convolution, then fused batch norm, in JITed kernel where possible
- Run on fused layers configured for Resnet50
- Weighted sum of the layers is representative of FWD pass performance but not identical
- Single socket on an Intel® Scalable Xeon® 8180 processor with turbo mode disabled
- Batch size 224

Type	Count	C	K	H	W	R	S	stride	padh	padw
NoBNRelu	1	3	64	224	224	7	7	2	3	3
NoBNRelu	1	64	256	56	56	1	1	1	0	0
BNRelu	3	64	256	56	56	1	1	1	0	0
NoBNRelu	1	64	64	56	56	1	1	1	0	0
BNRelu	3	64	64	56	56	3	3	1	1	1
BNReluEltwise	2	256	64	56	56	1	1	1	0	0
NoBNRelu	1	256	512	56	56	1	1	2	0	0
BNReluEltwise	1	256	128	56	56	1	1	2	0	0
BNRelu	4	128	128	28	28	3	3	1	1	1
BNRelu	4	128	512	28	28	1	1	1	0	0
BNReluEltwise	3	512	128	28	28	1	1	1	0	0
NoBNRelu	1	512	1024	28	28	1	1	2	0	0
BNReluEltwise	1	512	256	28	28	1	1	2	0	0
BNRelu	6	256	256	14	14	3	3	1	1	1
BNRelu	6	256	1024	14	14	1	1	1	0	0
BNReluEltwise	5	1024	256	14	14	1	1	1	0	0
NoBNRelu	1	1024	2048	14	14	1	1	2	0	0
BNReluEltwise	1	1024	512	14	14	1	1	2	0	0
BNRelu	3	512	512	7	7	3	3	1	1	1
BNRelu	3	512	2048	7	7	1	1	1	0	0
BNReluEltwise	2	2048	512	7	7	1	1	1	0	0

Layers mapped onto ResNet-50 topology require these configurations

Results

- Best case 21% speedup for JIT folding over Unfused kernel
- Most of the benefit (18%) comes from fused batch normalization demonstrated in [2]
- Additional 2% benefit from folding
- Should investigate networks with more compute which would allow for more overlap opportunities
- Another possibility is to recompute batch norm in the backward pass, which will allow us to avoid writing the batch-normed inputs



[1] D. Jung, W. Jung, , B. Kim, S. Lee, W. Rhee, and J. H. Ahn, "Restructuring batch normalization to accelerate cnn training," 2018.

[2] E. Georganas, S. Avancha, K. Banerjee, D. Kalamkar, G. Henry, H. Pabst, and A. Heinecke, "Anatomy of high-performance deep learning convolutions on simd architectures," in *Accepted to Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '18. IEEE Press, 2018.