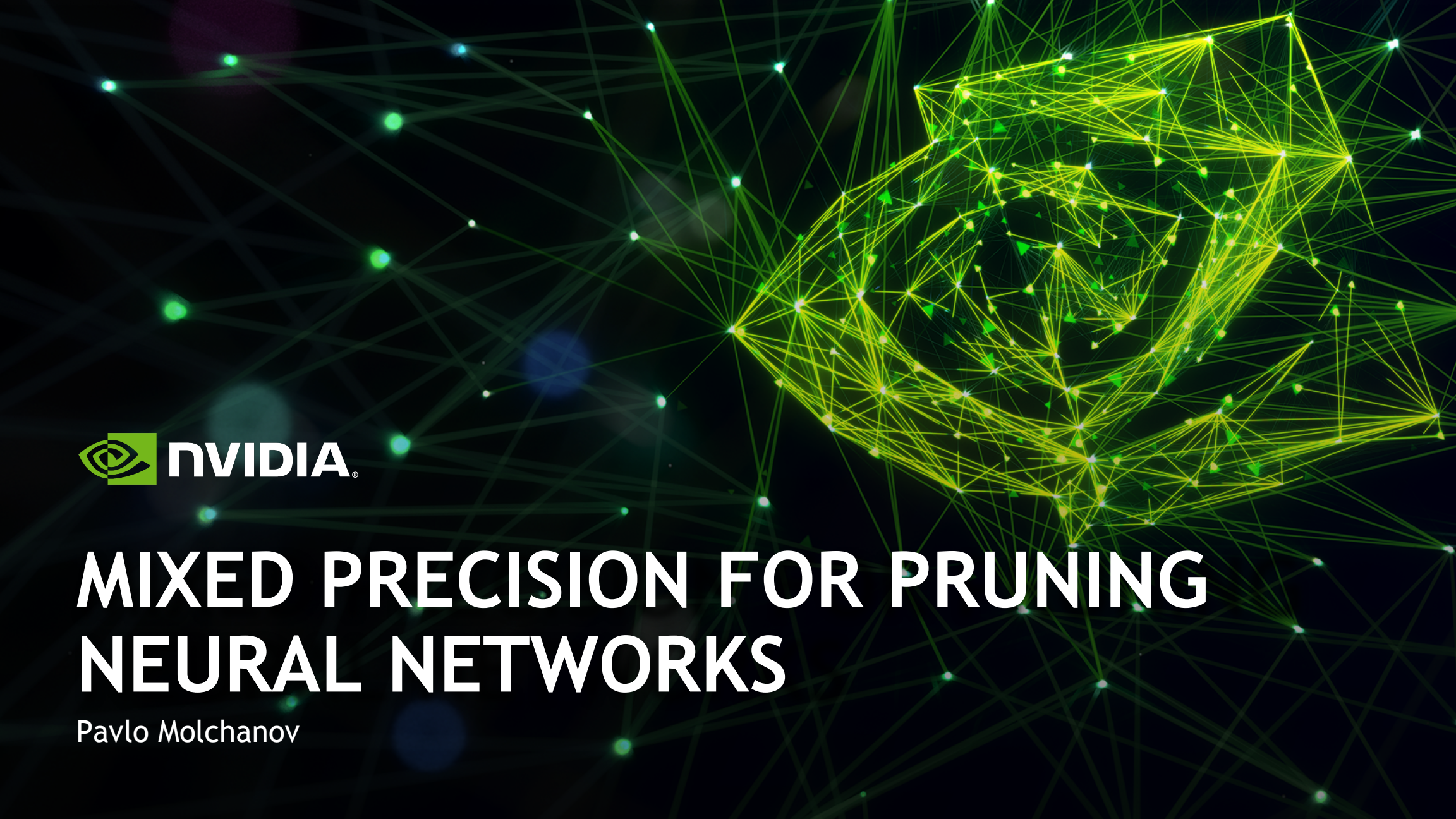
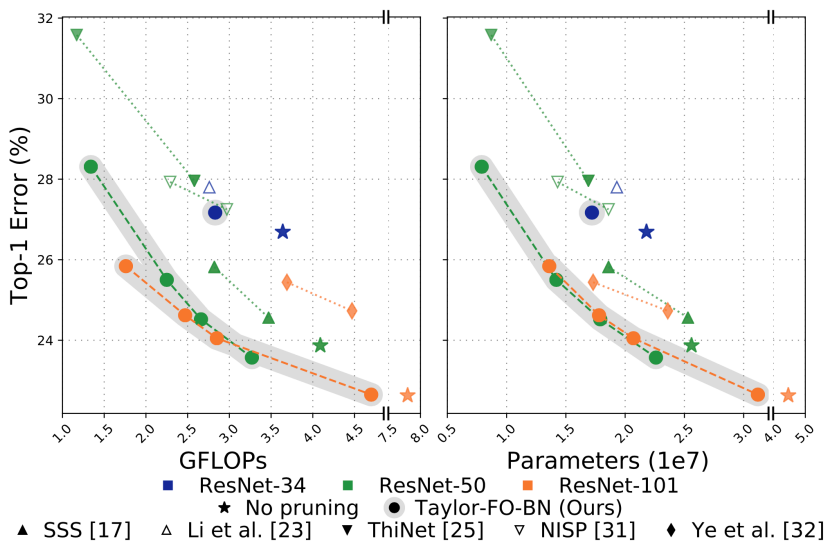


MIXED PRECISION FOR PRUNING NEURAL NETWORKS

Pavlo Molchanov

APPLICATION

- Pruning feature maps from pretrained neural network
- Based on CVPR2019 work:
  - *Importance estimation for Neural Network Pruning*
    Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, Jan Kautz
    NVIDIA



PRUNING

via gates

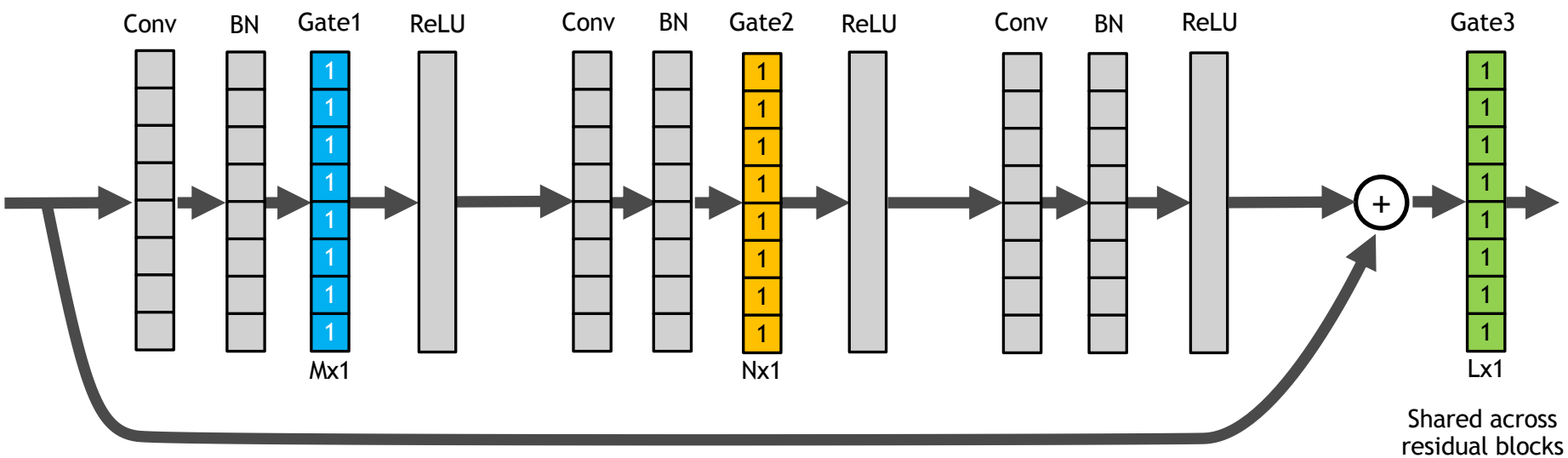
- Adding non learnable layer with constant weight of **1**
- Multiplies feature channel with **1**
- During backpropagation accumulates gradient

Importance estimate:

$$\mathcal{I}_m^{(1)}(\mathbf{W}) = g_m^2$$

Where g is a gradient

ResNet block:



CHANGES REQUIRED

```
if args.amp:
    model, optimizer = amp.initialize(
                model, optimizer,
                opt_level=args.opt_level,
                loss_scale=args.static_loss_scale)

# training loop
for batch_idx, (data, target) in enumerate(train_loader):
    optimizer.zero_grad()

    output = model(data)

    loss = criterion(output, target)

    if args.amp:
        with amp.scale_loss(loss, optimizer) as scaled_loss:
            scaled_loss.backward()
    else:
        loss.backward()

    optimizer.step()

    pruning_engine.do_step(loss=loss.item(), optimizer=optimizer)
```

RESNET-101

Approximating importance

The Spearman correlation coefficient = the Pearson correlation coefficient between the rank variables:

For a sample of size *n*, the *n* raw scores X_i, Y_i are converted to ranks $\operatorname{rg} X_i, \operatorname{rg} Y_i$, and r_s is computed from:

$$r_s = \rho_{\operatorname{rg}_X, \operatorname{rg}_Y} = \frac{\operatorname{cov}(\operatorname{rg}_X, \operatorname{rg}_Y)}{\sigma_{\operatorname{rg}_X} \sigma_{\operatorname{rg}_Y}}$$

Bottleneck convs only:

AMP optimization level	Per layer	Global
fp32 (amp-O0)	0.7458	0.9316
fp16-ampO1 (mixed precision)	0.7485	0.9319
fp16-ampO2 (mixed precision)	0.7499	0.9320

All layers including skip connections:

AMP optimization level	Per layer	Global
fp32 (amp-O0)	0.7470	0.9199
fp16-ampO1 (mixed precision)	0.7495	0.9210
fp16-ampO2 (mixed precision)	0.7512	0.9213

From https://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient

RESNET-101

Experiment details

- ResNet-101 pruning via 50% feature removal including skip connections
- Total unique gates 20096, we remove 9632, groups of 32
- Finetuning loss: standard cross-entropy loss
- Pruning metric: squared gradient
- Hardware 4 x NVIDIA V100 GPU
- Batch size is 4x64 = 256
- Finetuning for 25 epochs, lr = 0.0015, lr decay each 10 steps
- Removing 10k neurons during first 187 mini-batches, pruning is finished within the first epoch
- Based on PyTorch ImageNet training example

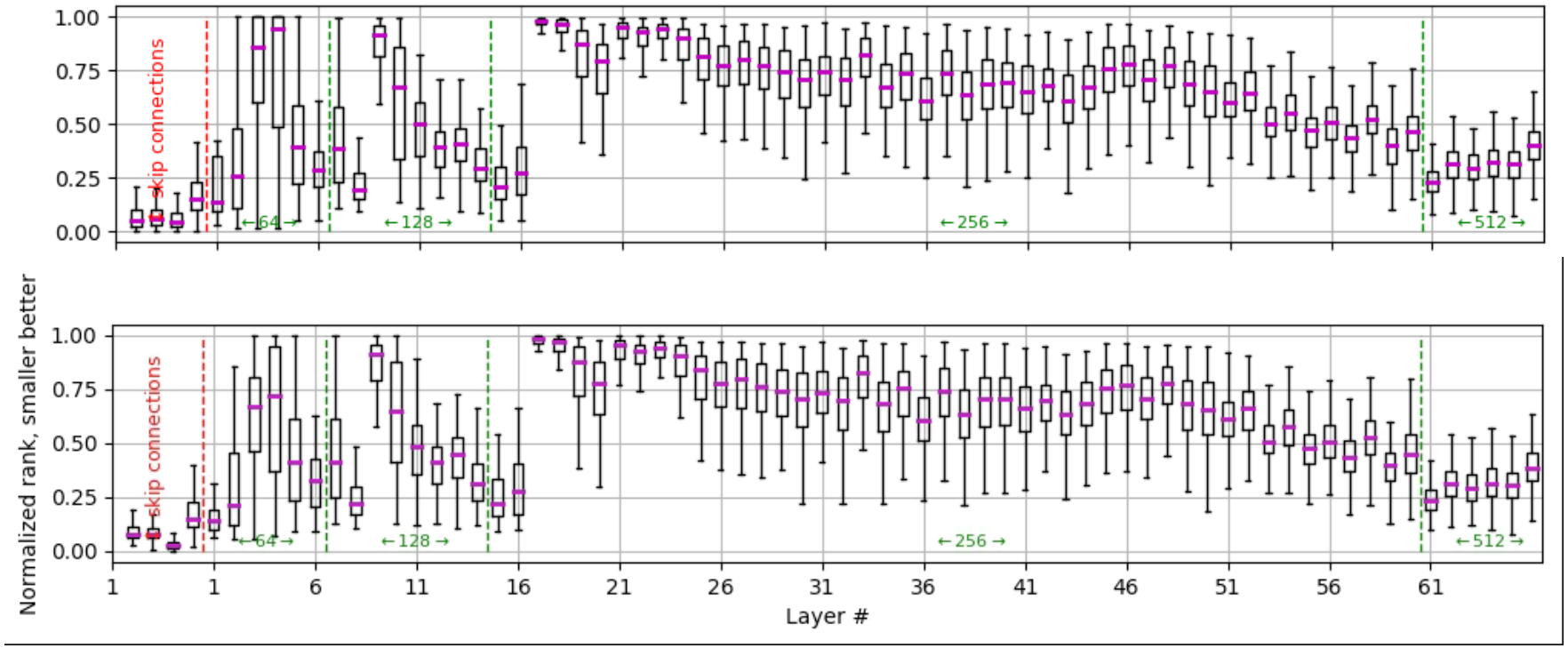
RESNET-101

Obtained speedups

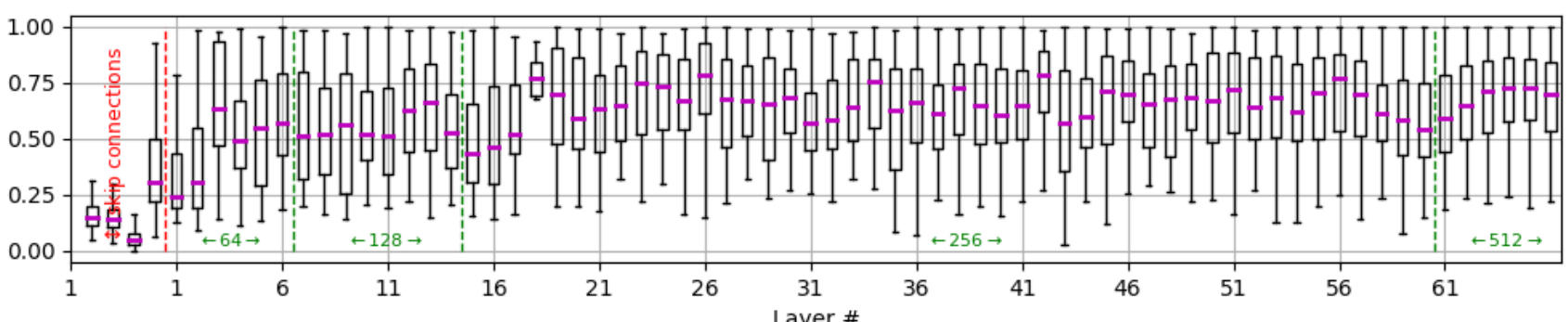
- Pruning 50% feature maps from ResNet-101
- Evaluated on ImageNet dataset

Model	AMP optimization level	Time (25 epochs)	Accuracy (top-1)	GFLOPs /img	Parameters, x1e7
ResNet-101 full			77.37%	7.80	4.47
ResNet-101 pruned 50%	fp32 (amp-O0)	13h 14m	75.73%	2.54	1.96
	fp16-ampO1 (mixed precision)	9h 7m **(1.35x speedup)**	75.81%	2.54	1.94
	fp16-ampO2 (mixed precision)	6h 42m **(1.98x speedup)**	75.87%	2.53	1.94

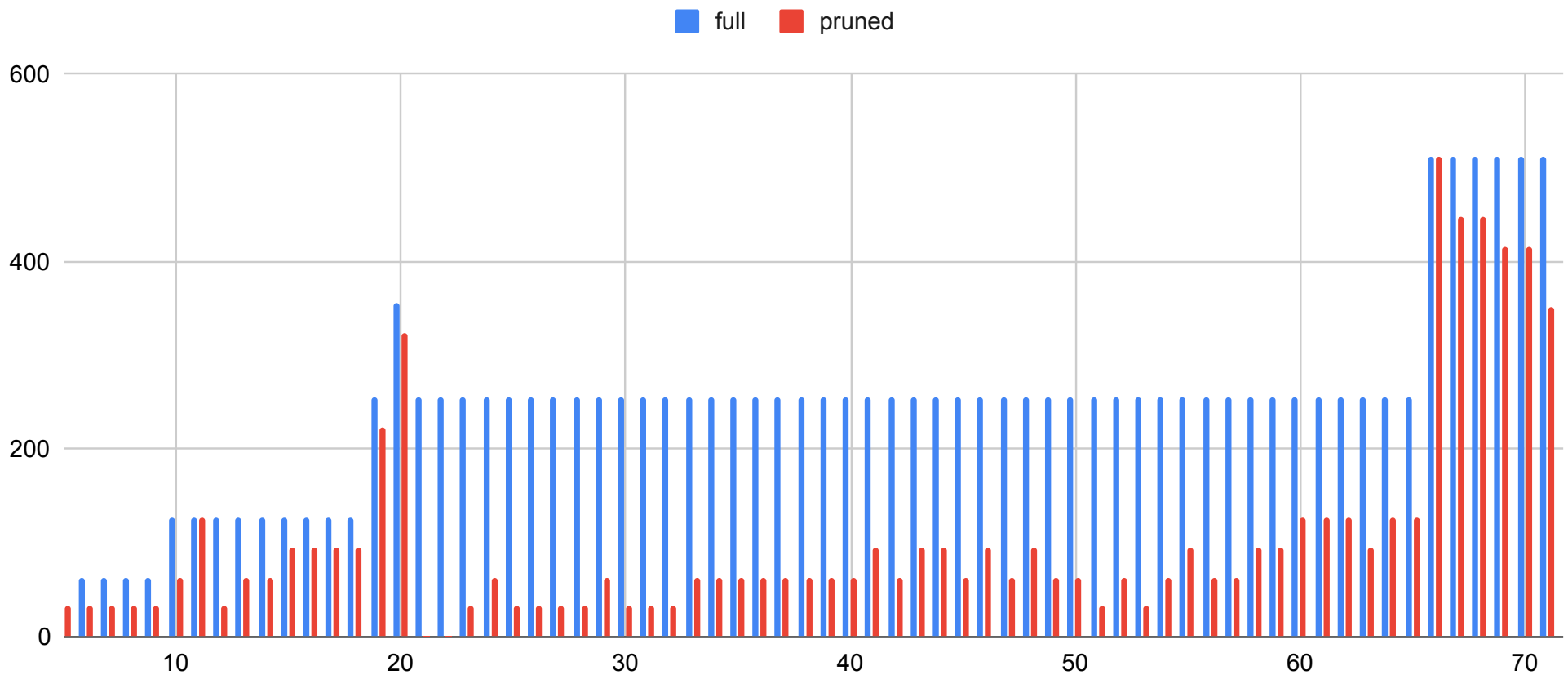
Before pruning



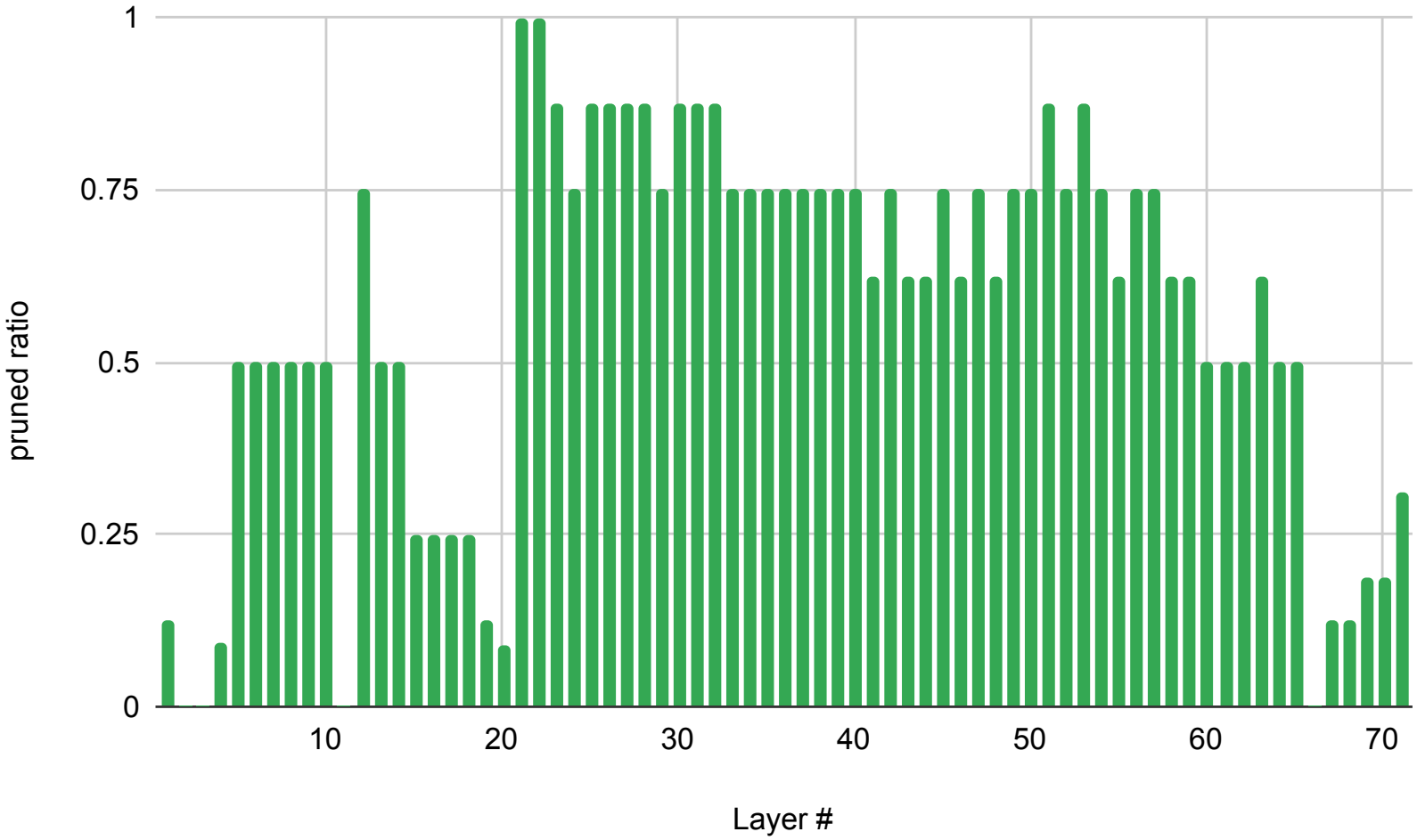
After pruning



RESNET-101



RESNET-101



RESNET-50 V 1.5

Experiment details

- ResNet50 v1.5 pruning. Adapted highly efficient code from NVIDIA examples: https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/RN50v1.5
- Original model trained for 90 epochs with cosine annealing learning rate
- NVIDIA DGX v1: 4xV100 GPUs
- Pruning 3200 unique feature maps from total of 7552, via groups of size 32
- 30 epochs, cosine lr, lr=0.1024, loss scale 128
- Batch size is 128x4 and virtually scaled to 1024

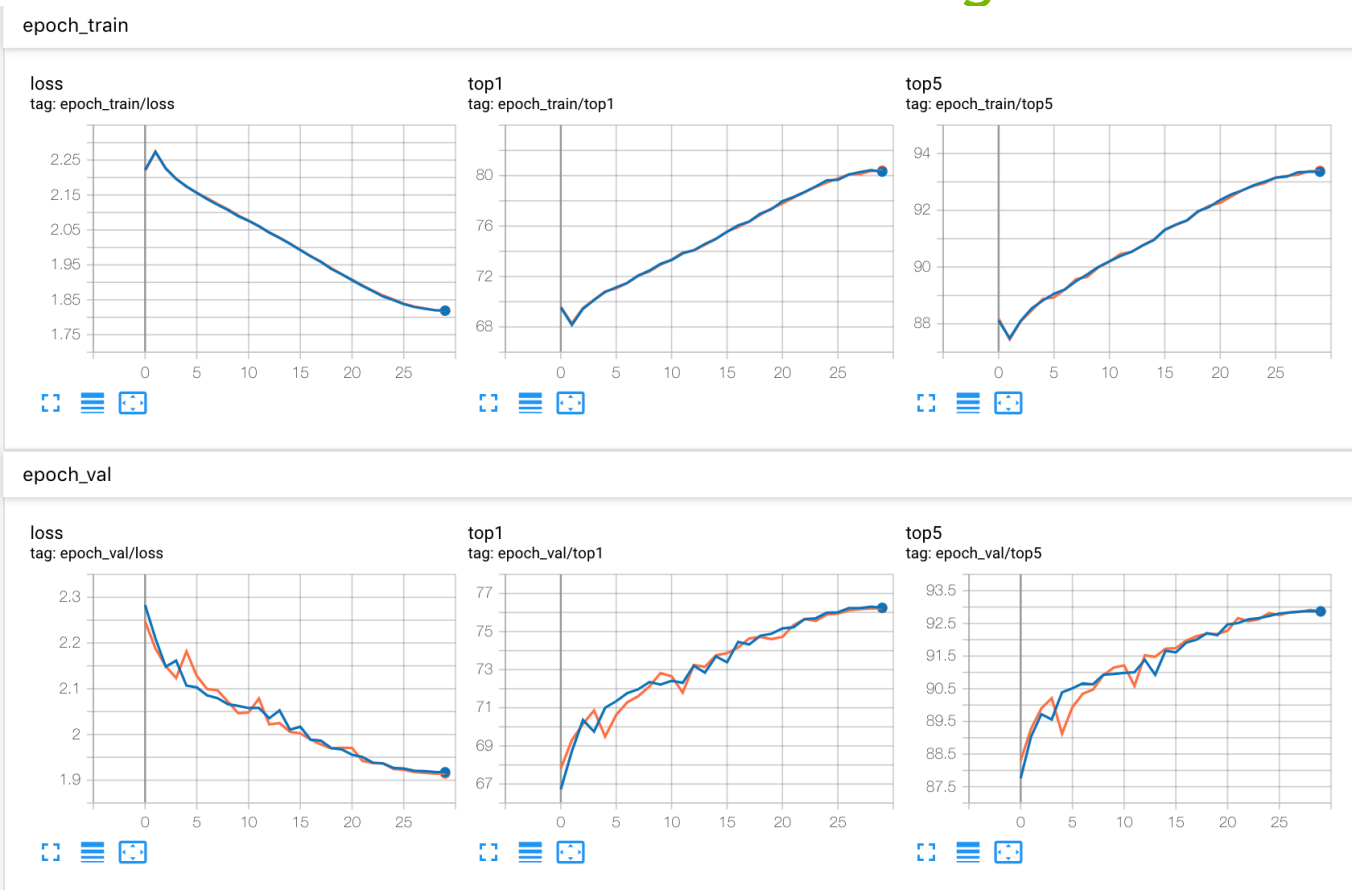
RESNET-50 V 1.5

Obtained speedups

AMP optimization level	Time	Accuracy (top-1)	GFLOPs/img	Parameters, x1e7
Original model		77.26%	4.071	2.56
fp32 (amp-O0)	8h 05m	76.25%	2.33	1.49
fp16-ampO1 (mixed precision)	4h 26m **(1.82x speed-up)**	76.17%	2.33	1.49
fp16-ampO2 (mixed precision)	4h 22m **(1.85x speed-up)**	76.23%	2.33	1.48

RESNET-50 V 1.5

Training curves



Blue - fp32

Orange - amp-O2 (fp16)

IMPORTANT OBSERVATIONS

- Holds for fp16-ampO2 all time, and fp16-ampO1 when pruning parameter is not part of optimizer.
- Changes to parameters need to be performed after optimizer.step() as it will rewrite any changes with master weights
- Dynamic vs static loss scaling. When gradients are scaled back? Better to keep static loss scaling for pruning.

TAKEAWAYS

- Pruning in mixed-precision is equally as effective as pruning in fp32
- Speed up in turnaround time is 1.35-1.98x
- Up 2x less memory consumption on GPU with mixed-precision


NVIDIA