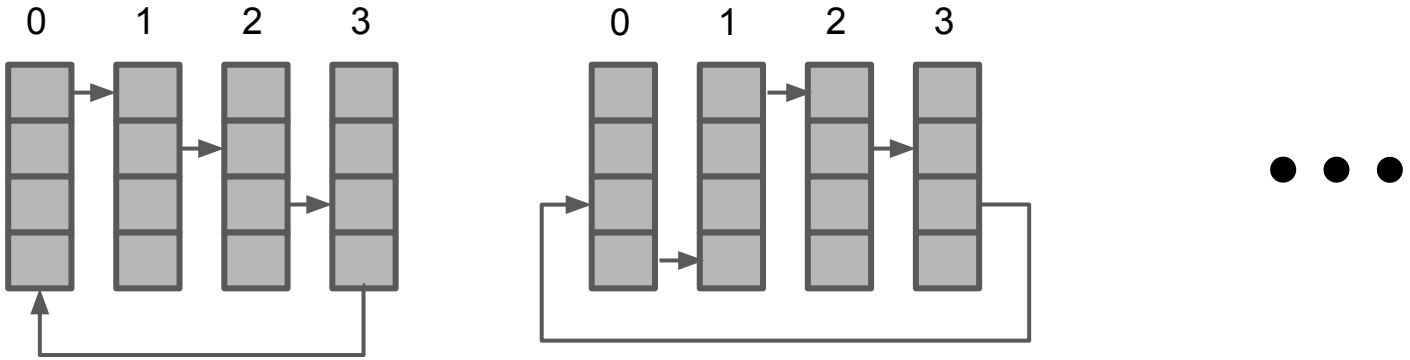
Characterization of float non-associativity in NCCL

Ring algorithm

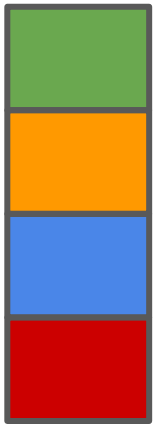


step 1

step 2

This is a **single** ring.
Each chunk starts **at different device** (due to algorithm) and thus goes through **different paths**: e.g. first chunk in {0-1-2-3}; second chunk in {1-2-3-0}, etc.

Ring algorithm



In the end, each device would have the same data. Chunks here are color-coded according at which device the summation started.

- Starts from 0 => {0-1-2-3}
- Starts from 1 => {1-2-3-0}
- Starts from 2 => {2-3-0-1}
- Starts from 3 => {3-0-1-2}

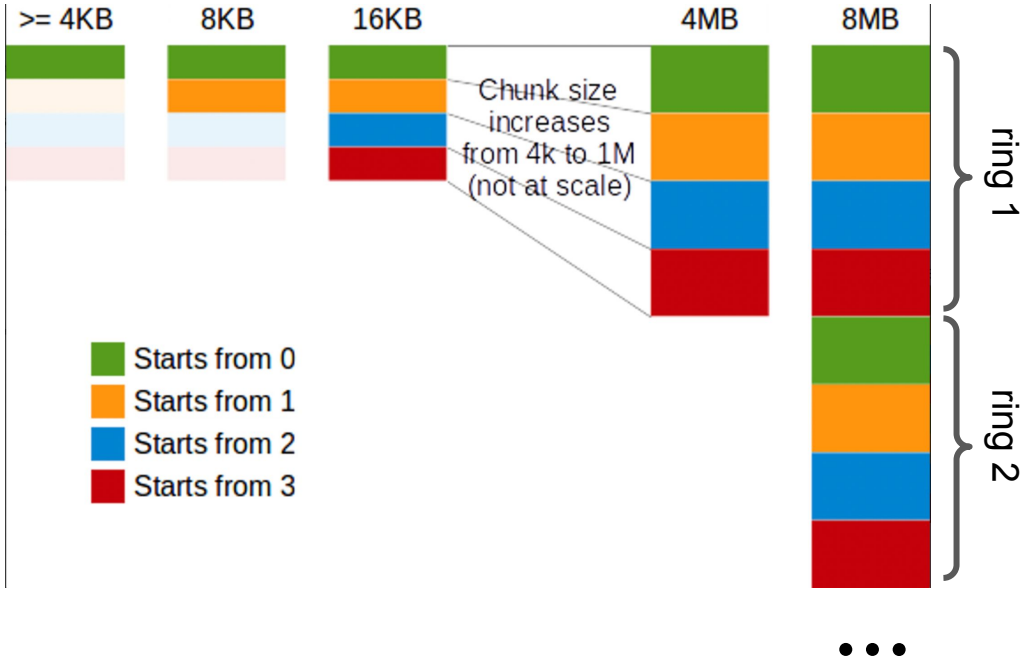
NCCL

In NCCL, order by which a particular data element will be reduced is defined by its **offset**, relative from beginning of the message. Message is defined as an array called into AllReduce kernel, e.g.

ncclAllReduce(message);

Chunk size in NCCL is *dynamic*, it can change from 4KB to a maximum of 1MB. Which offset is reduced in which order is determined by: number of devices, indices of devices, and message size, but the rules are **deterministic.** (Meaning that if you run the same thing on a same system with the same message size, you will always get the same output).

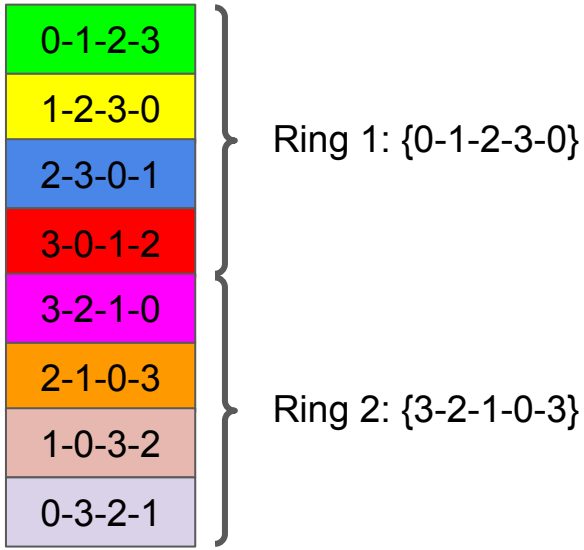
NCCL: How offsets are determined?



- Messages smaller than 4KB - are not divided to chunks, and thus reduced in the same order.
- Then for larger messages, number of chunks is increased, until there are N chunks.
- For larger messages, chunk size is increased, up until the maximum of 1MB.
- For even larger messages, ***multiple rings*** are used, with chunks in each of them. **The order is different in each ring**.

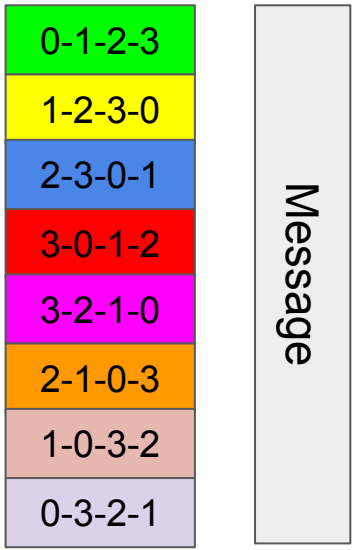
NCCL

Now imagine, we have two rings: [0->1->2->3->0] and [3->2->1->0->3], i.e. simple bidirectional ring. Assuming that the message is large enough to occupy both rings, the order of offsets would look like below (in each chunk, the order in which it is reduced is shown). Treat this image as **a map of offsets**



Float Associativity Problem

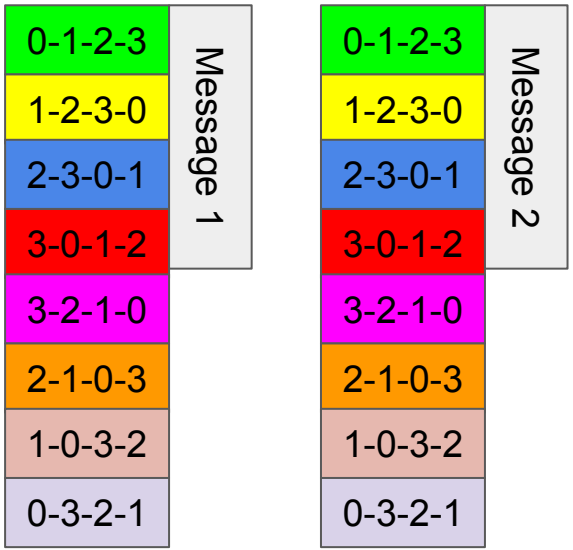
If we have a message of fixed size, and just repeat AllReduce **on the same system** lots of times, **output will be the same**, because each data element's offsets are the same.



ncclAllReduce(message);
ncclAllReduce(message);
ncclAllReduce(message);
…

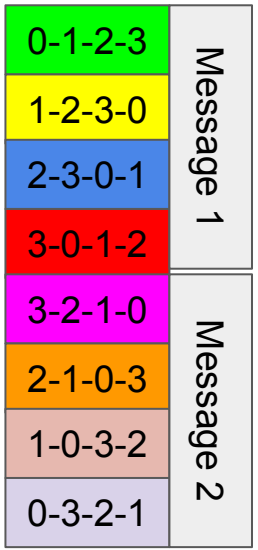
Float Associativity Problem

However, if we combine two messages or do something of this sort, **outputs will be different**, because each data element's **offsets would have changed**.



ncclAllReduce(message1);
ncclAllReduce(message2);

VS



ncclAllReduce([message1,message2]);

Takeaways

- Float non-associativity does not matter when running the AllReduce on a same system (with same N of devices) and with a message of the fixed size and with the same number of rings.

- Float non-associativity matters when combining or splitting messages, for example: allreducing gradients one by one vs bucketing them into larger segments won't be numerically identical.

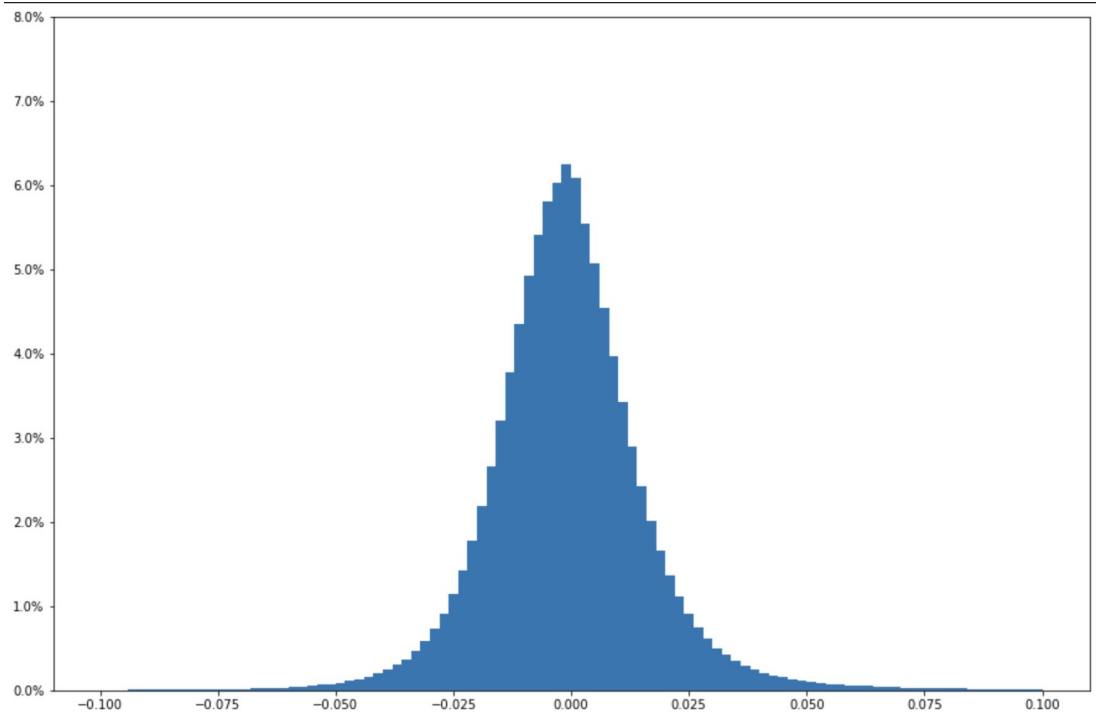
- Float non-associativity matters when number of used devices is different, or when topology is different and NCCL constructs rings differently.

- Float non-associativity could be observed in ML workloads by varying the number of rings, thus for some gradients the order of computation will change.

Exploring distribution of numbers observed in real ML workloads

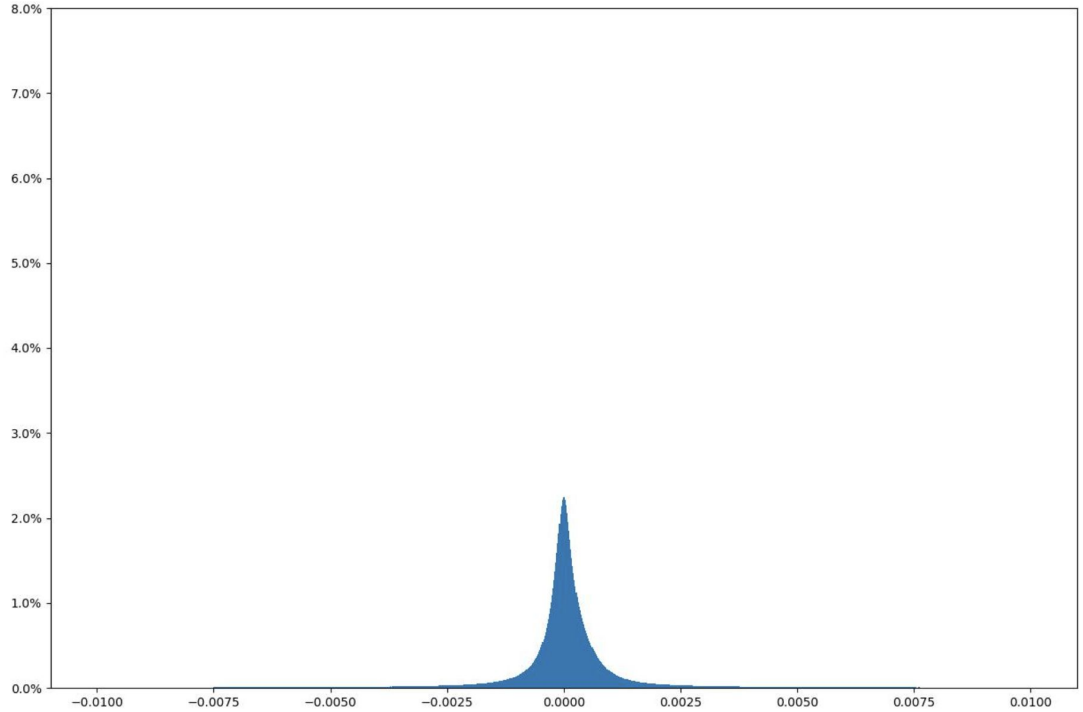
ResNet-50

Distribution of trained weights: Gaussian with μ=-0.00036, σ = 0.02048



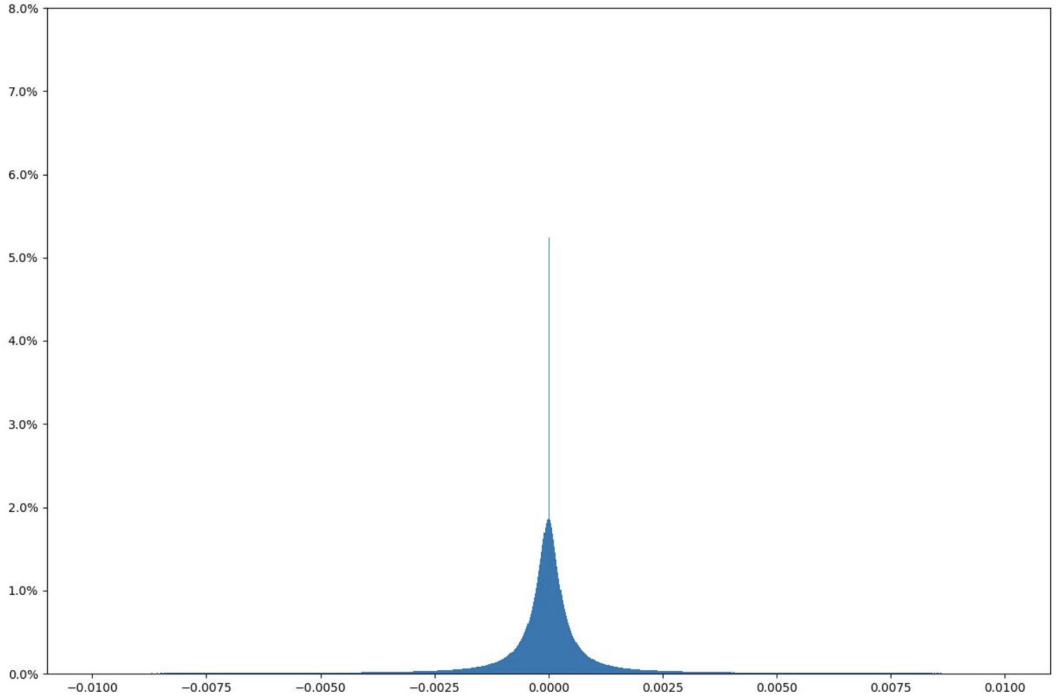
ResNet-50

Distribution of gradients: *Iteration 1:* μ=-8.72e-6, σ = 0.02667



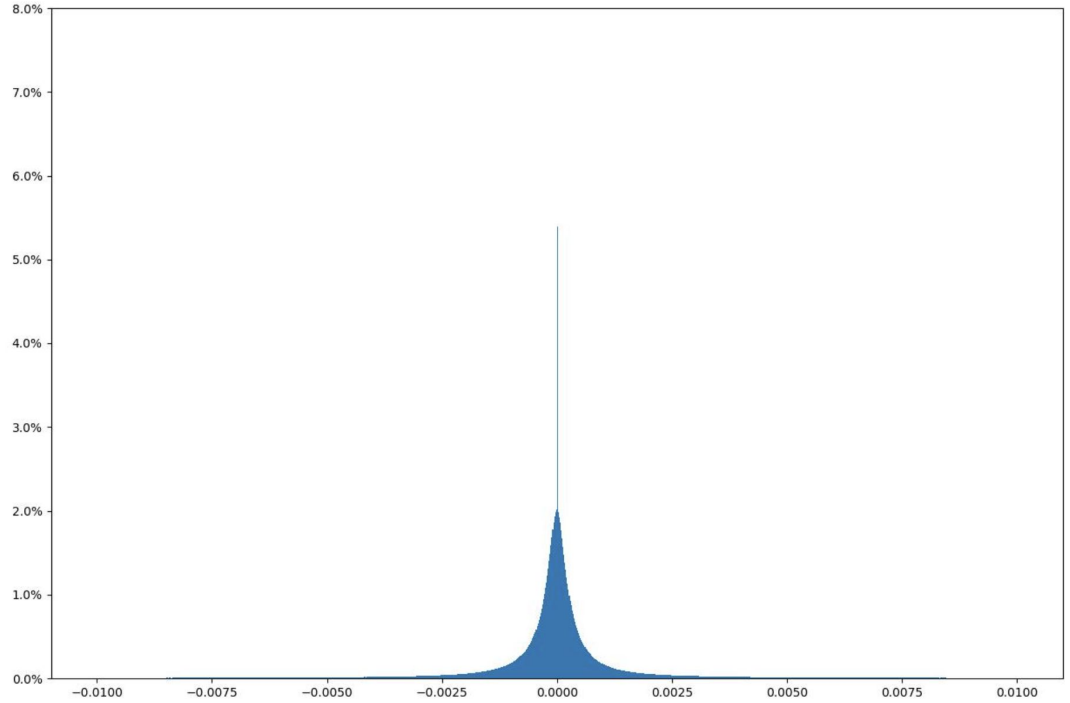
ResNet-50

Distribution of gradients: *Iteration 5:* μ=-2.37e-5, σ = 0.01271



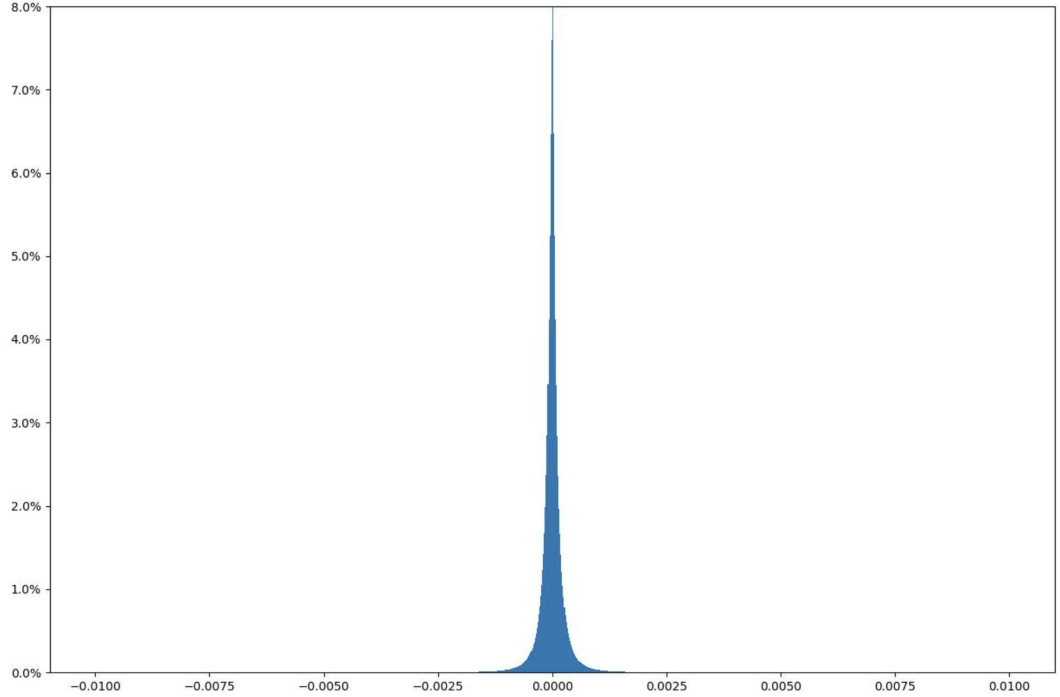
ResNet-50

Distribution of gradients: *Iteration 10:* μ=-4.23e-6, σ = 0.0457



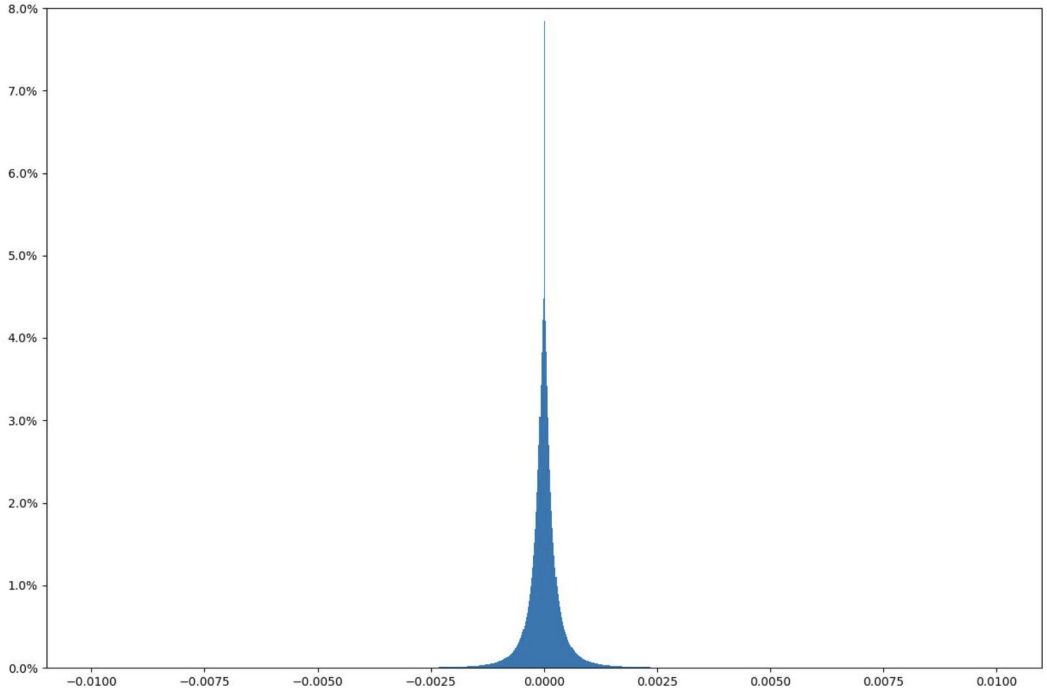
ResNet-50

Distribution of gradients: *Iteration 100:* μ=5.08e-7, σ = 0.000336



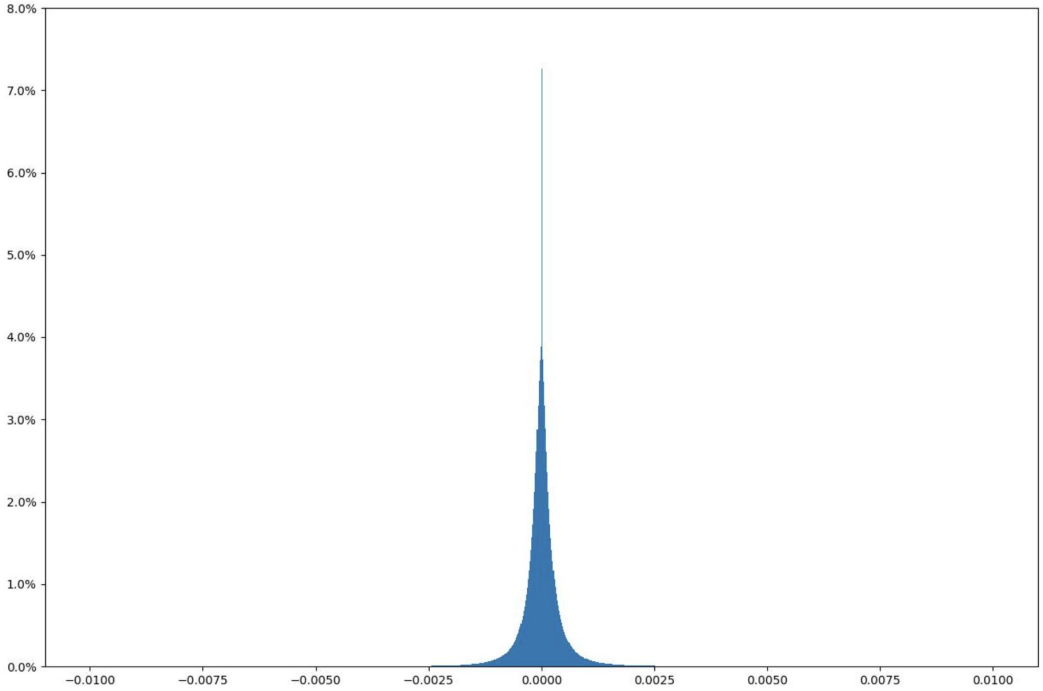
ResNet-50

Distribution of gradients: *Iteration 200:* μ=1.15e-6, σ = 0.000506



ResNet-50

Distribution of gradients: *Iteration 300:* μ=2.899e-6, σ = 0.000511. ***Using this one in math analysis.***



Takeaways:

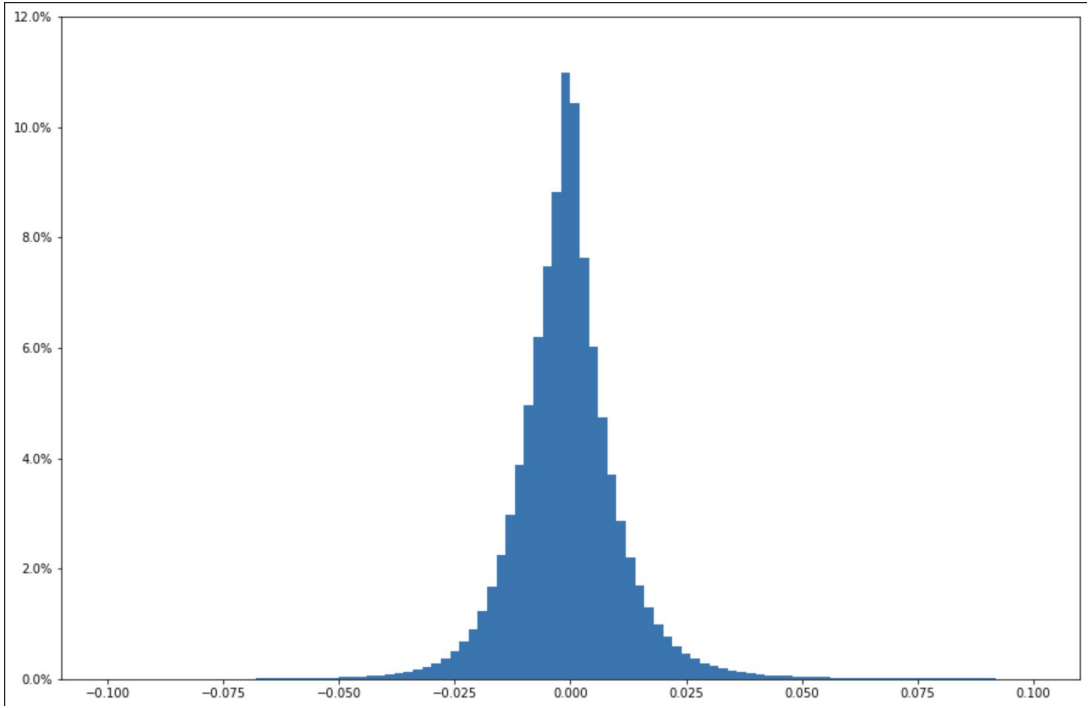
- Distribution of gradients is very similar on all GPUs, with very close mean and std at each iteration. Values at the same index are usually very close in magnitude

- Distribution of weights is always the same on all GPUs, since the same weights are maintained on all devices

- Distribution of gradients is centered around zero and is very close to zero

ResNet-152

Distribution of trained weights: Gaussian with μ=-0.00039, σ=0.0141



Analysis using math script

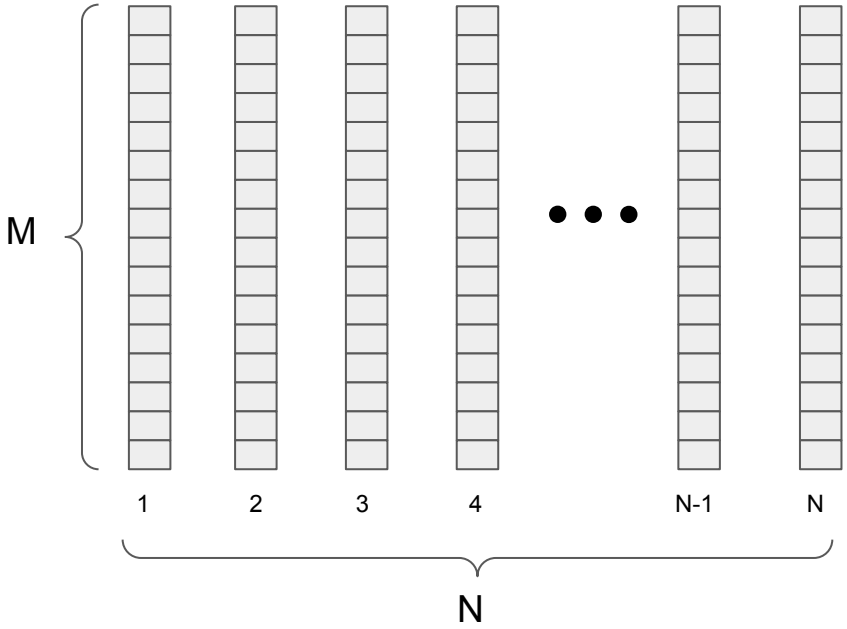
Defining difference metric

We will have N vectors, each having numbers summed in different order. We need to define a simple metric, a single number between 0 and 1, that will show us how “different” these vectors are. Previously, I did some things with standard deviation and etc, but those are poorly defined and confusing.

I propose to use cosine similarity. Cosine similarity is defined between two vectors, but it can be extended to multiple vectors by calculating **average pairwise cosine similarity**. Then we will have a single metric that will allow us to observe difference caused by summation in different orders and we can do plots and such.

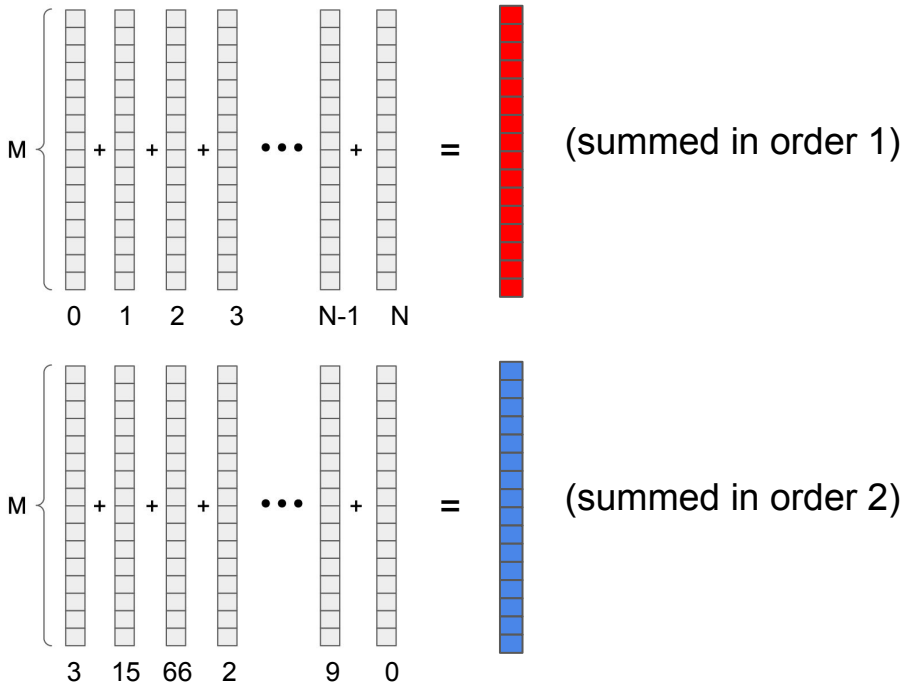
Methodology

1) Generate random values according to the gradient distribution.



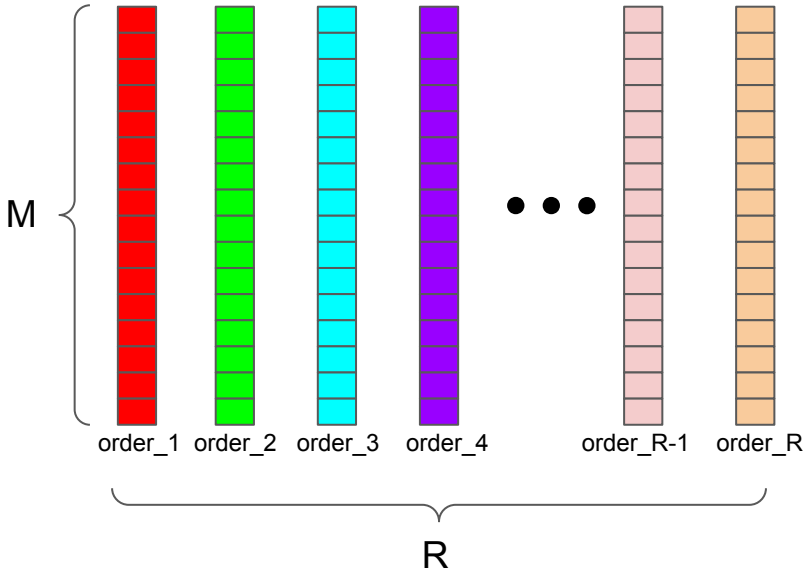
Methodology

2) Sum the values along the N-axis, in R <u>random non-repeating</u> orders.



Methodology

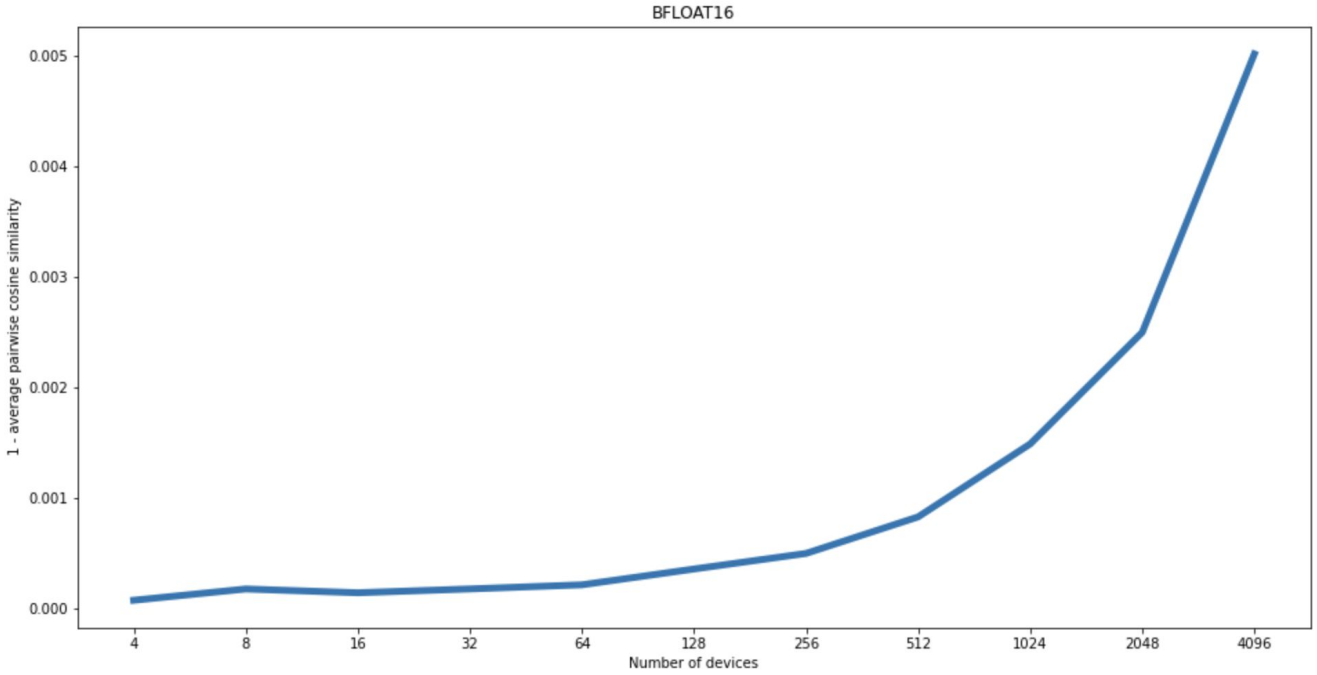
3) In the end, we are going to have R versions of sums, each summed in different order. Compute average pairwise cosine similarity.



(1 - average pairwise cosine similarity)

Metric of difference

Results

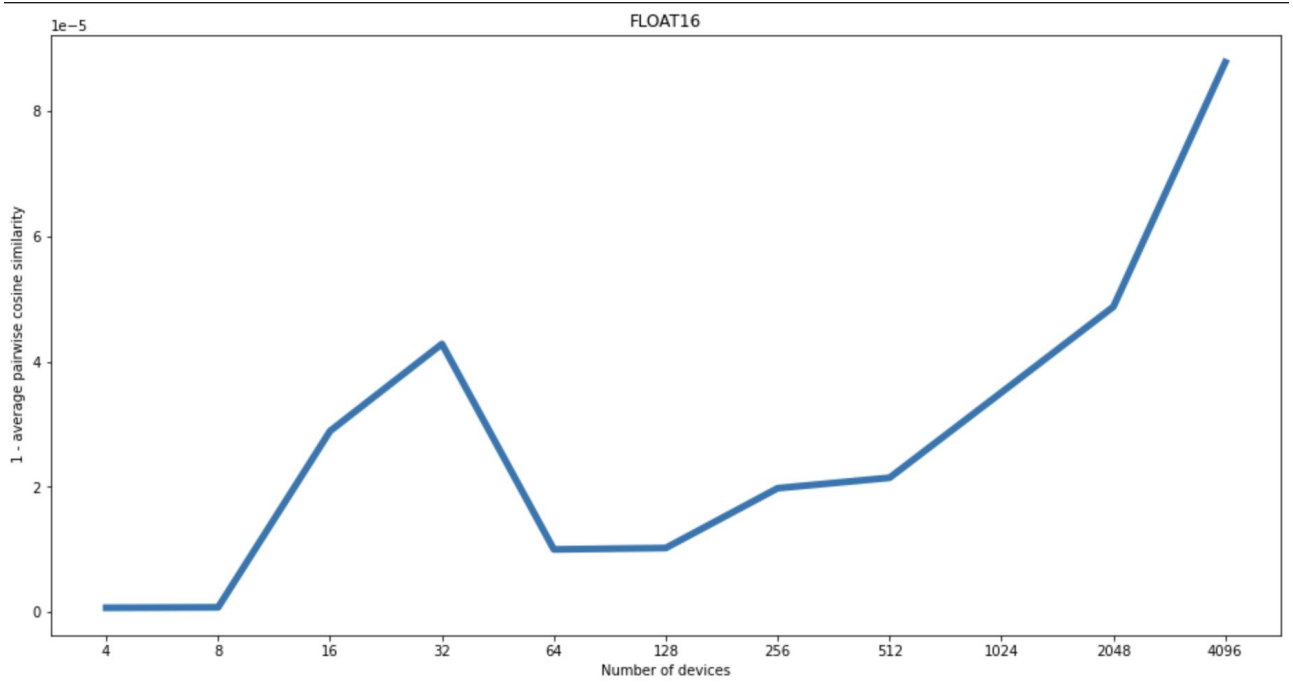


This is result with numbers according to ResNet50 gradient distribution.

With bfloat16, difference due to summation order grows with N.

X-axis in logarithmic scale

Results



Same trend with float16, but difference is orders of magnitude smaller.

Difference with float16 and with bfloat16 is more or less insignificant with N=4

Takeaways:

- Difference due to order of summation grows linearly with N. This is confirmed.
- With bfloat16 difference is much more pronounced. With bfloat16 and N>8 we should observe really significant difference in training due to summation order, according to the math analysis.
- Type of the distribution has little to no effect, from what i’ve observed so far. I am double checking it now, and can add the plots as well, but they don’t look that different from those above
- I need to define the experiment with hierarchical, what we are testing and what do we want to see?

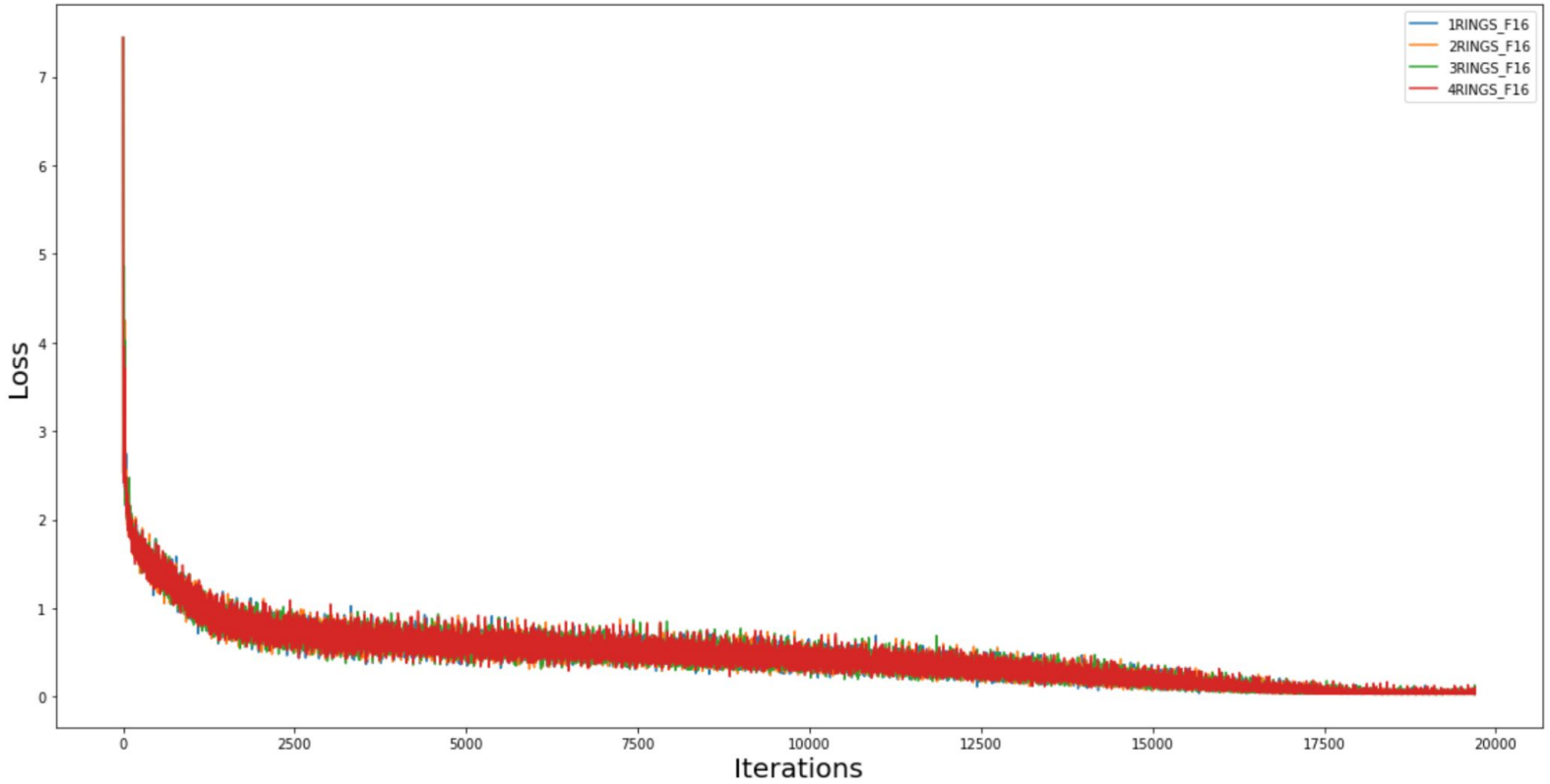
Experiments on ML workloads

ResNet-50

Training hyperparameters, for reproducibility purposes:

- CIFAR10 classification task
- random seed = 20214229
- cross entropy loss
- sgd optimizer
- lr = 1e-2, momentum = 0.9, L2 penalty = 1.5e-2
- CosineAnnealing learning rate scheduler with T=200
- batch_size = 128 **per GPU**
- epochs = 200

ResNet-50 -- float16. Loss convergence



ResNet-50 -- float16. Accuracies on the test set.

Determinism enabled

Rings	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	30.62%	76.62%	77.68%	84.38%	90.05%
2	30.45%	77.01%	79.48%	85.7%	89.59%
3	29.21%	78.32%	81.18%	86.38%	89.94%
4	30.46%	76.3%	80.91%	86.31%	89.89%

Metric of difference = `0.000116`

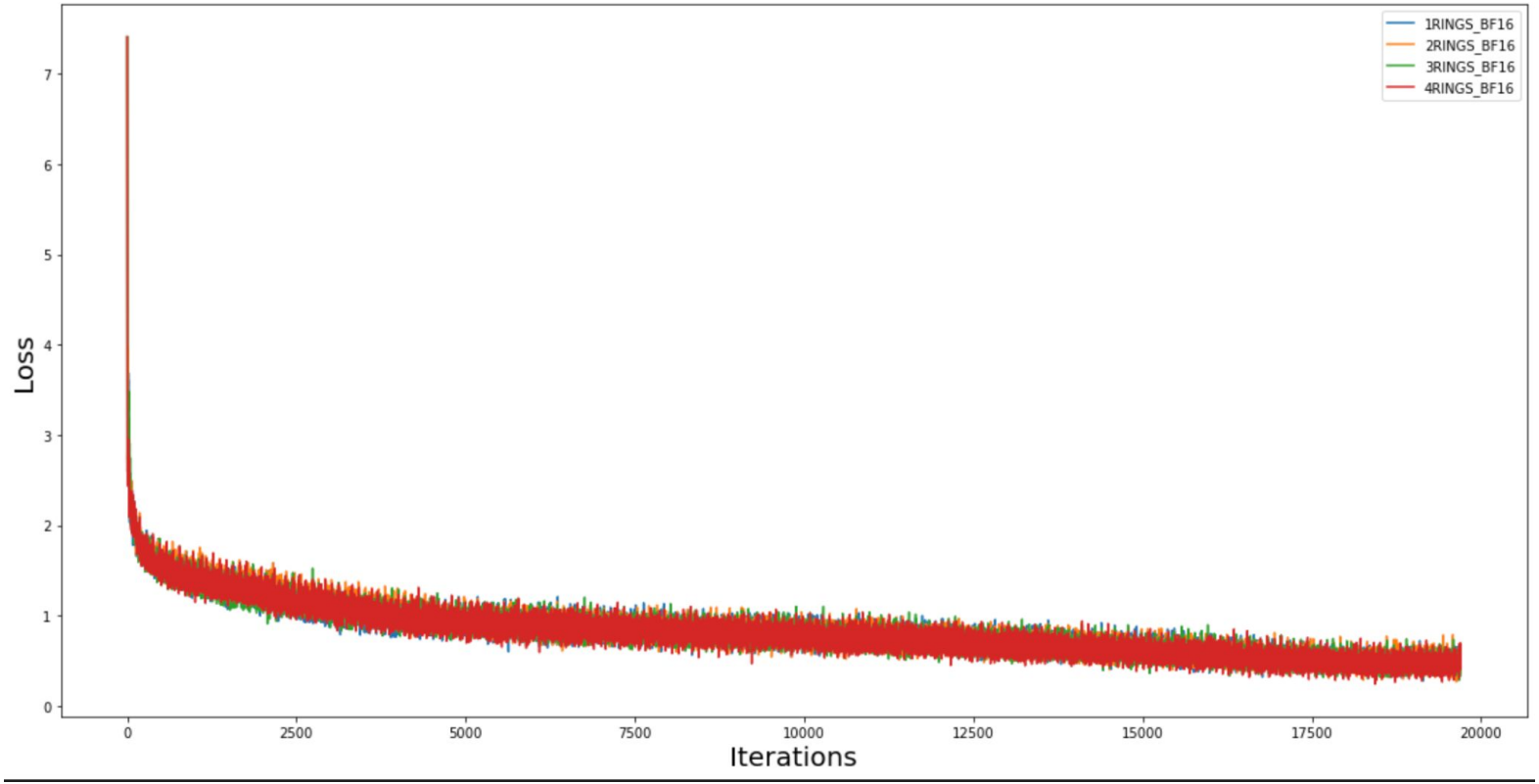
ResNet-50 -- float16. Accuracies on the test set.

Determinism disabled. Same #Rings = 4, but different runs

Runs	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	31.41%	76.62%	78.99%	86.18%	90.1%
2	29.05%	73.78%	79.55%	85.29%	89.8%
3	30.59%	75.59%	77.14%	85.33%	89.67%
4	30.91%	76.57%	81.17%	85.8%	89.76%

Metric of difference = `0.000136`

ResNet-50 -- bfloat16. Loss convergence



ResNet-50 -- bfloat16. Accuracies on the test set.

Determinism enabled.

Rings	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	32.04%	64.7%	69.55%	76.36%	81.07%
2	28.74%	68.24%	68.87%	74.82%	80.53%
3	31.53%	66.07%	71.31%	76.34%	81.9%
4	32.25%	67.71%	73.19%	74.48%	82.69%

Metric of difference = `0.000364`

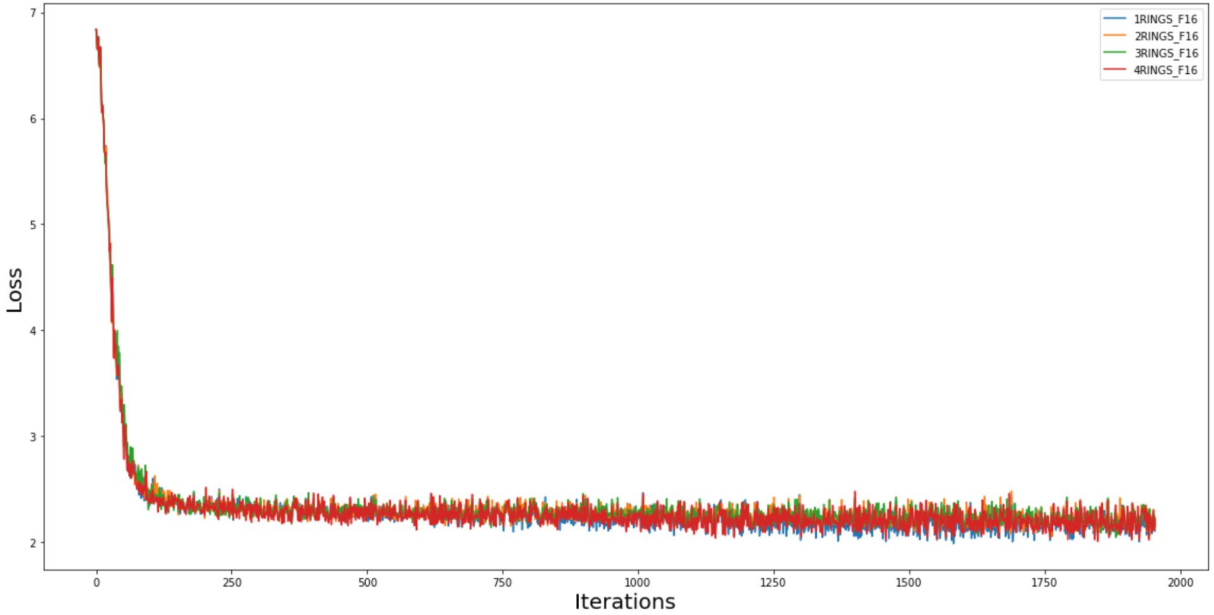
ResNet-50 -- bfloat16. Accuracies on the test set.

Determinism disabled. Same #Rings = 4, but different runs

Runs	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	%	%	%	%	%
2	%	%	%	%	%
3	%	%	%	%	%
4	%	%	%	%	%

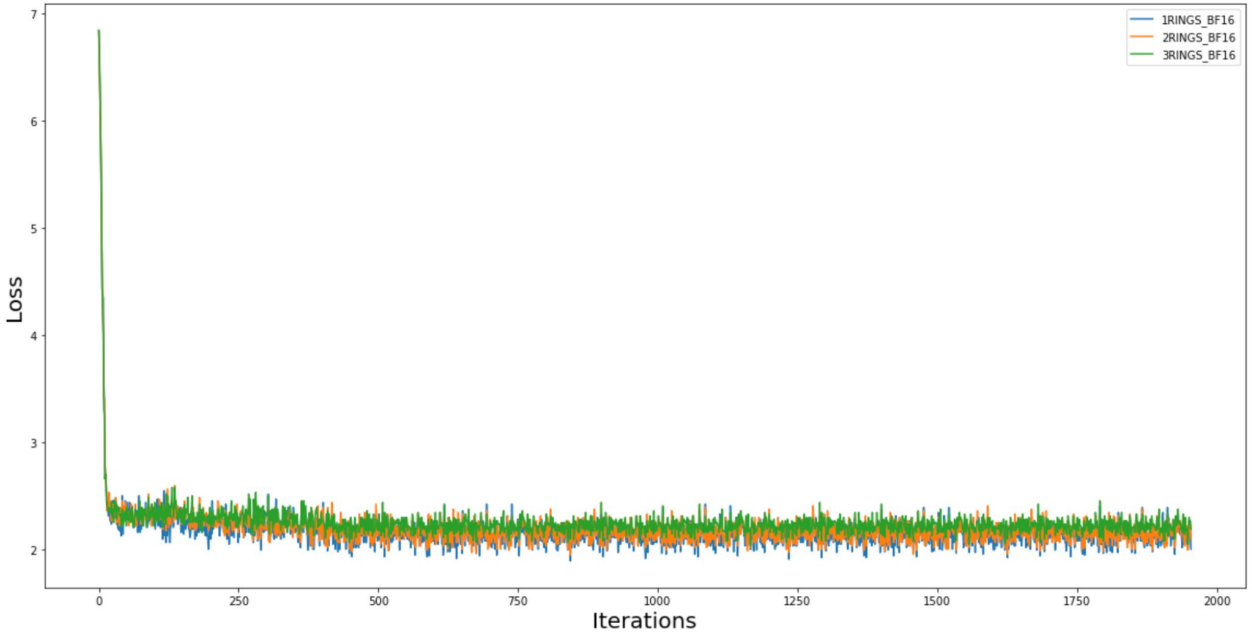
OLD PLOTS AND TABLES THAT NEED REVISION

ResNet-152 -- float16.



# Rings	Accuracy
1	19.96%
2	18.7%
3	18.3%
4	19.23%

ResNet-152 -- bfloat16. SGD optimizer



# Rings	Accuracy
1	21.87%
2	19.75%
3	17.14%
4	-

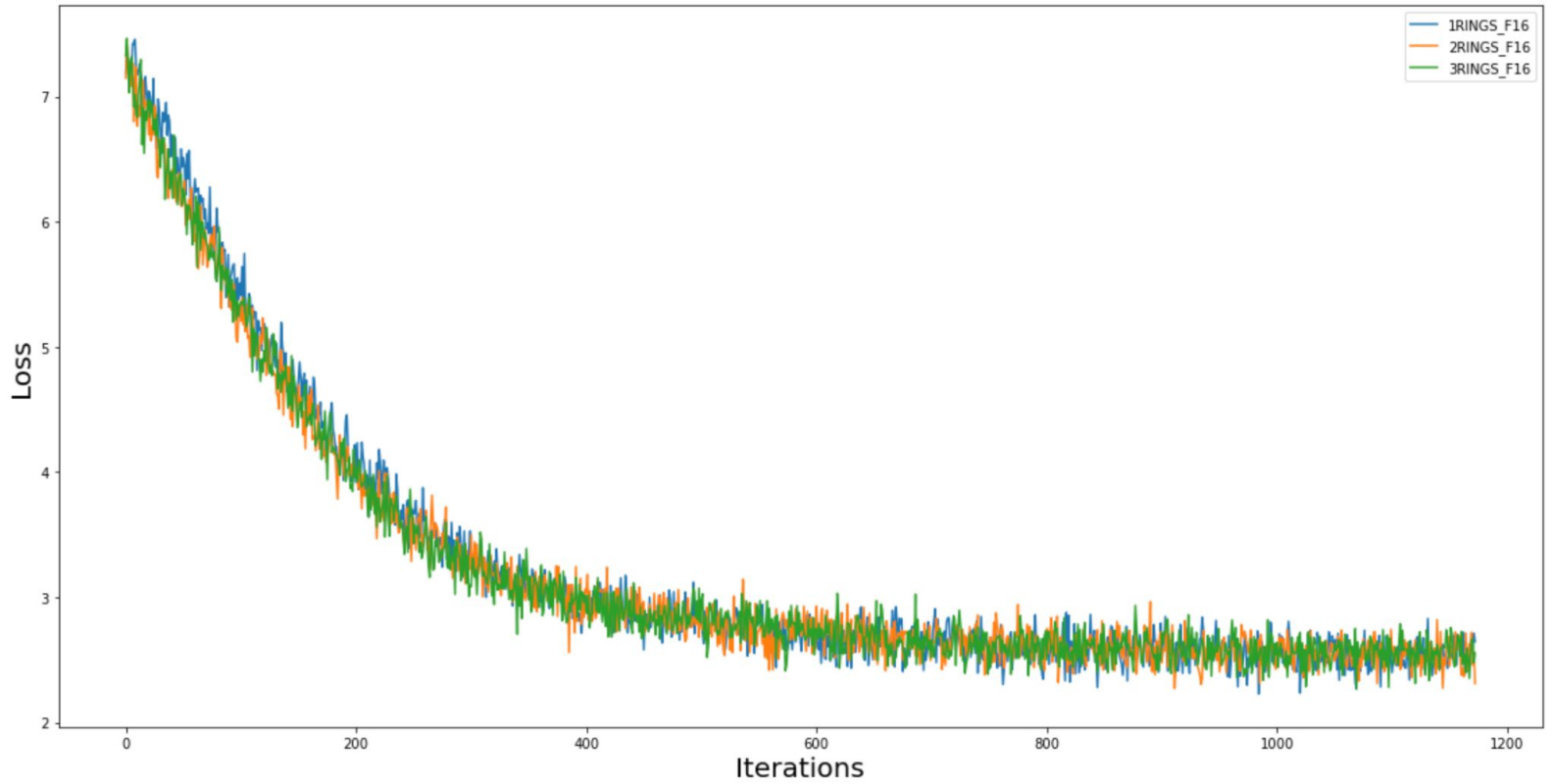
Results (M=1000, R=10000, values [0:1])

N =	16		64		256		512		1024		4096		16384	
	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16
avg. st. dev.	1e-4	2e-3	2e-4	3e-3	4e-4	6e-3	5e-4	4e-3	7e-4	0	1e-3	0	0	0
max st. dev.	2e-4	3e-3	3e-4	5e-3	5e-4	8e-3	7e-4	8e-3	9e-4	0	2e-3	0	0	0
av. range	6e-4	1e-2	1e-3	2e-2	2e-3	4e-2	3e-3	2e-2	4e-3	0	9e-3	0	0	0
max. range	1e-3	2e-2	2e-3	3e-2	4e-3	5e-2	5e-3	6e-2	7e-3	0	12e-3	0	0	0

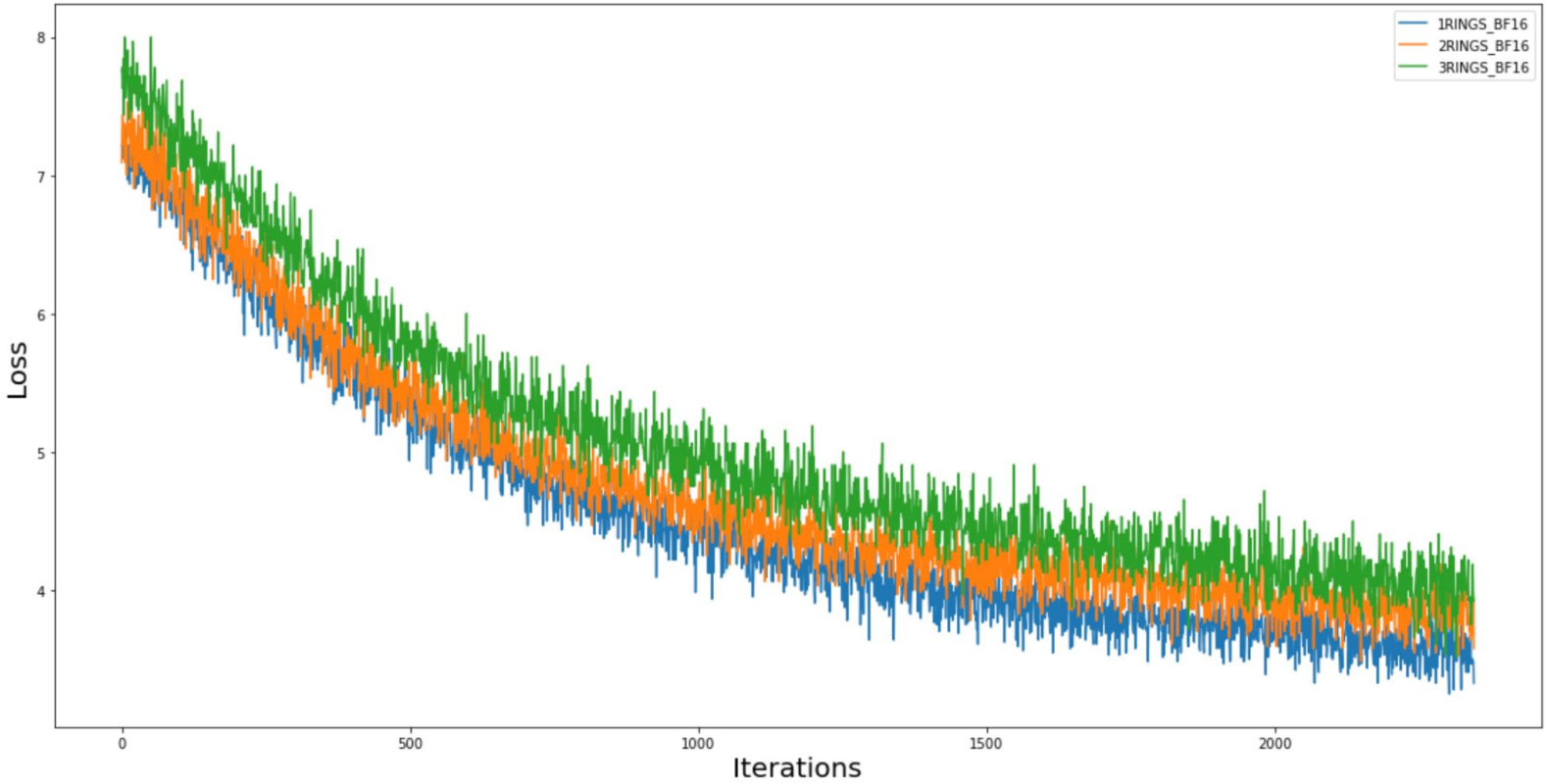
Results (M=1000, R=10000, values [-1:1])

N =	16		64		256		512		1024		4096		16384	
	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16	f16	bf16
avg. st. dev.	2e-4	2e-3	2e-4	2e-3	2e-4	2e-3	2e-4	2e-3	2e-4	2e-3	2e-4	2e-3	2e-4	2e-3
max st. dev.	7e-4	6e-3	6e-4	5e-3	7e-4	5e-3	5e-4	5e-3	6e-4	5e-3	5e-4	5e-3	5e-4	4e-3
av. range	2e-3	1e-2	2e-3	1e-2	2e-3	1e-2	2e-3	1e-2	2e-3	1e-2	2e-3	1e-2	2e-3	1e-2
max. range	4e-3	3e-2	4e-3	4e-2	5e-3	3e-2	4e-3	4e-2	4e-3	3e-2	4e-3	3e-2	5e-3	3e-2

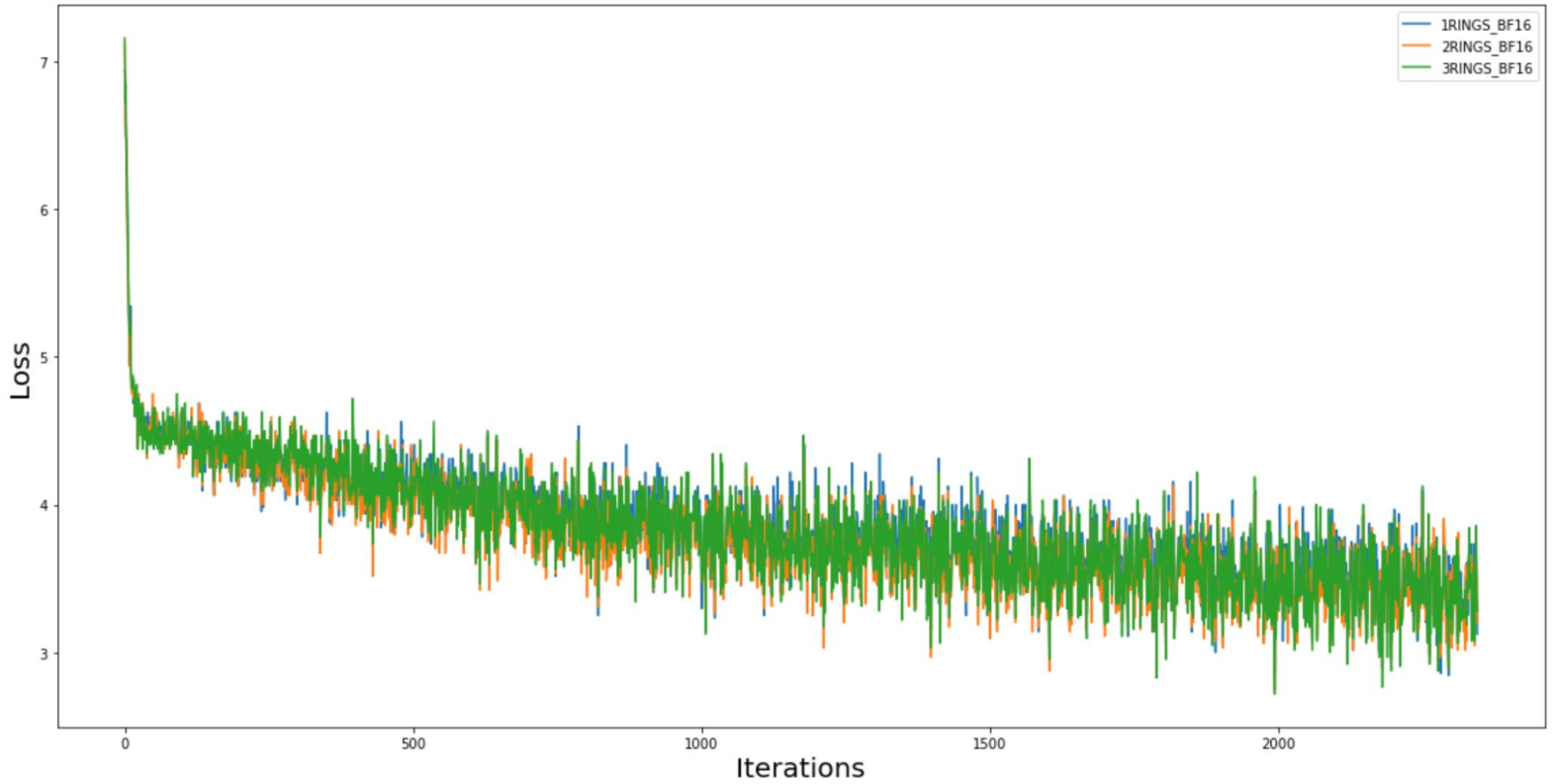
ResNet-152 -- float16. SGD optimizer



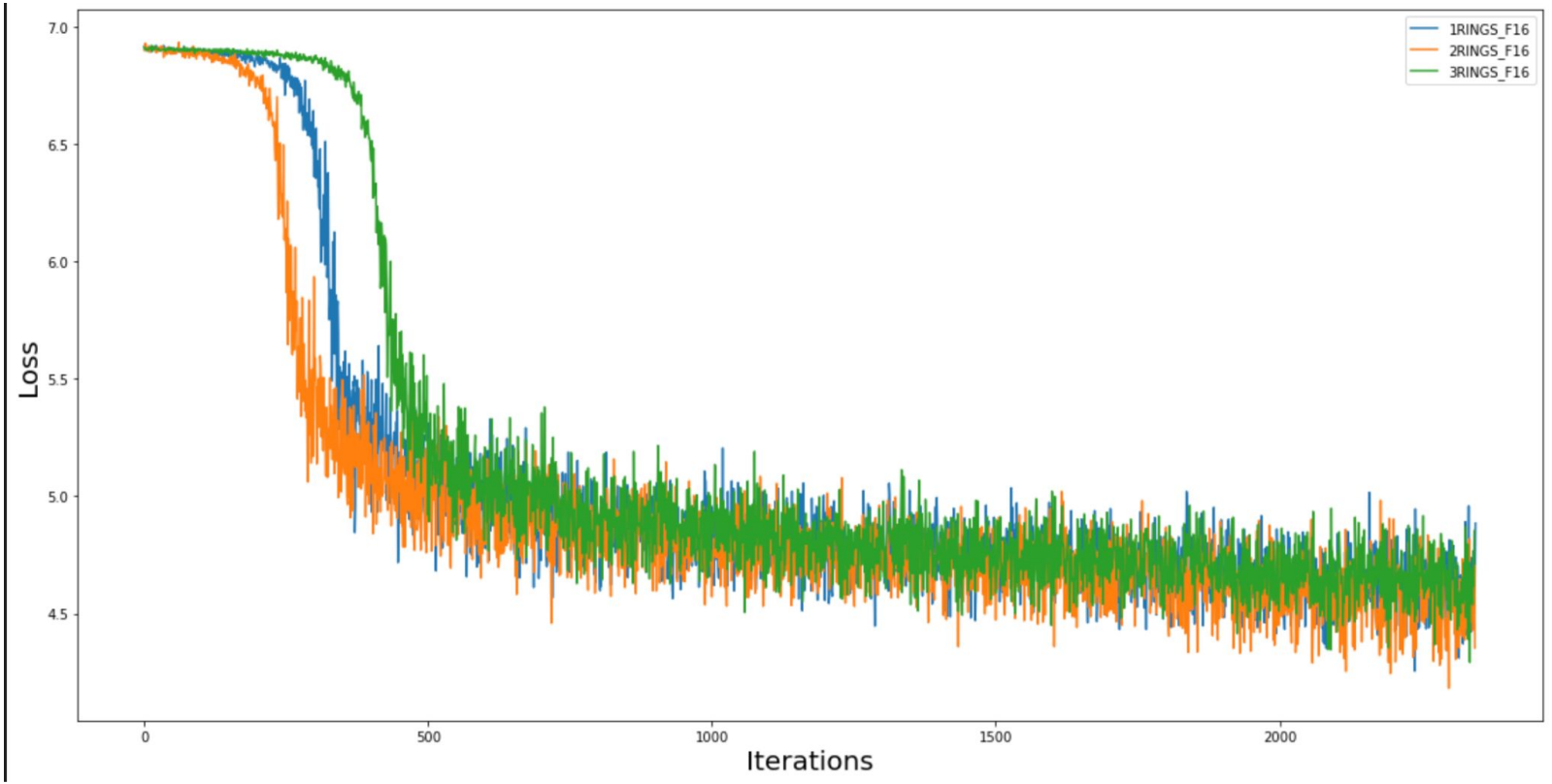
ResNet-152 -- bfloat16. SGD optimizer



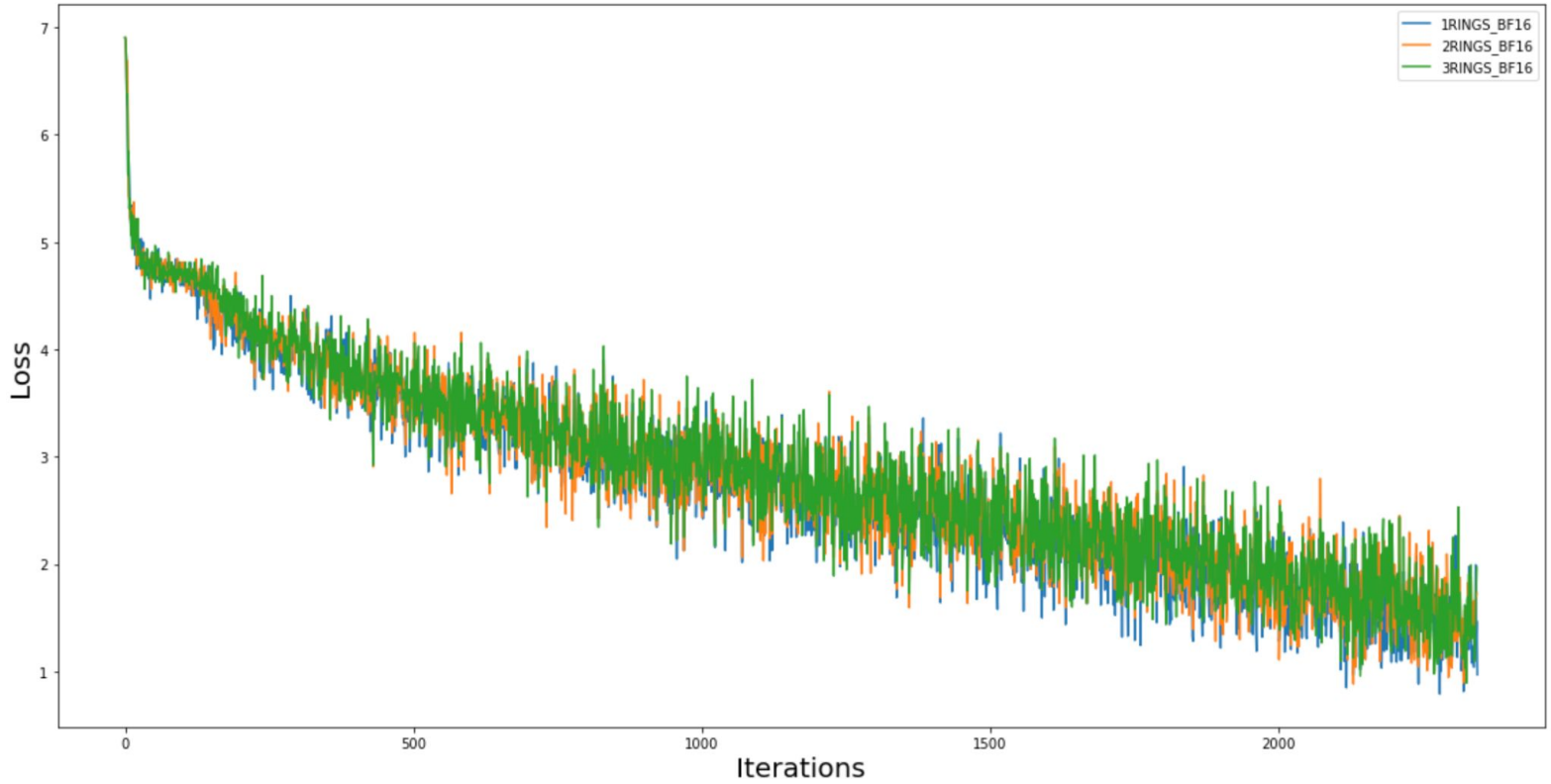
ResNet-152 -- bfloat16. Adam optimizer



VGG16 -- float16. SGD optimizer



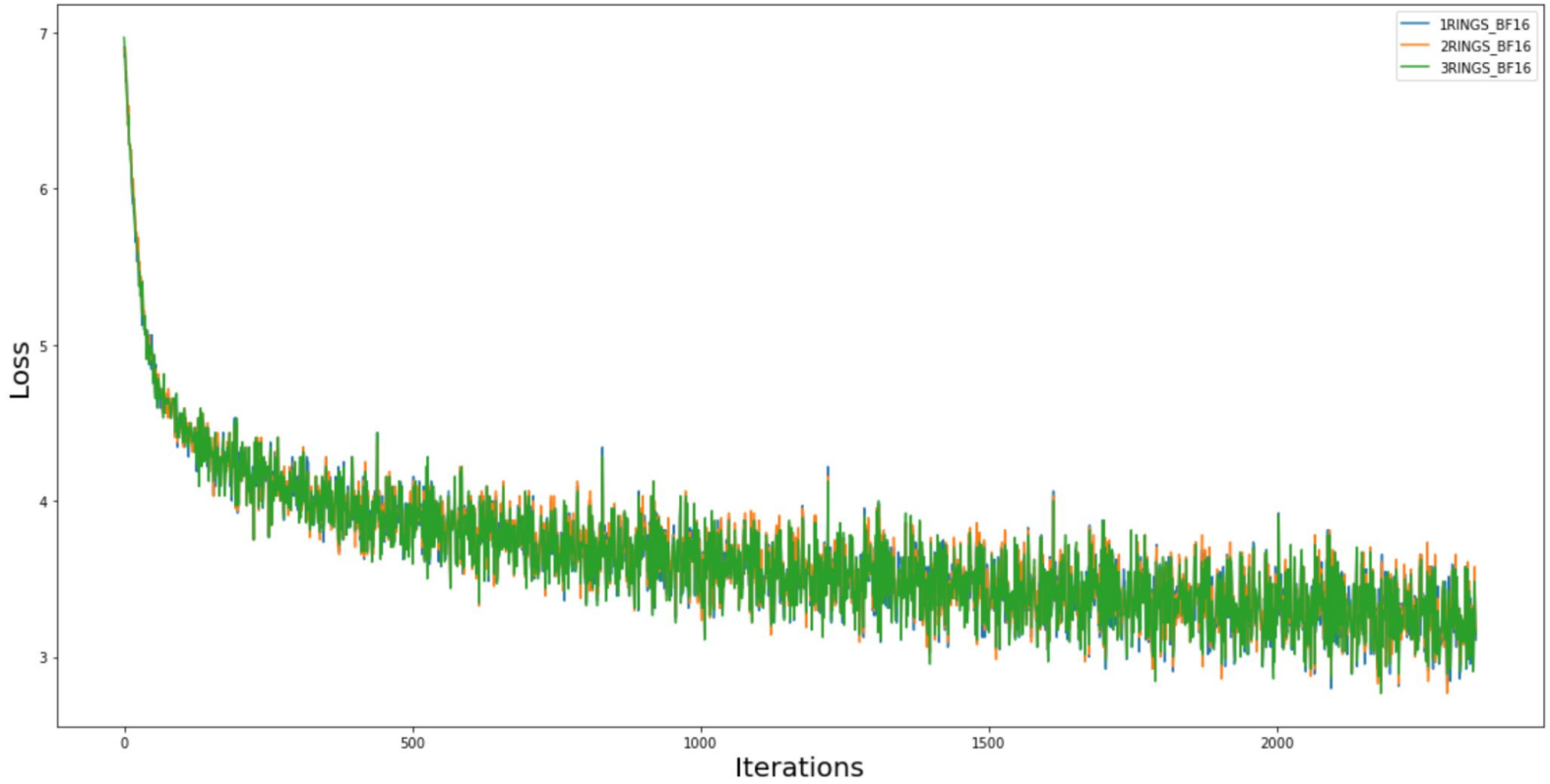
VGG16 -- float16. Adam optimizer



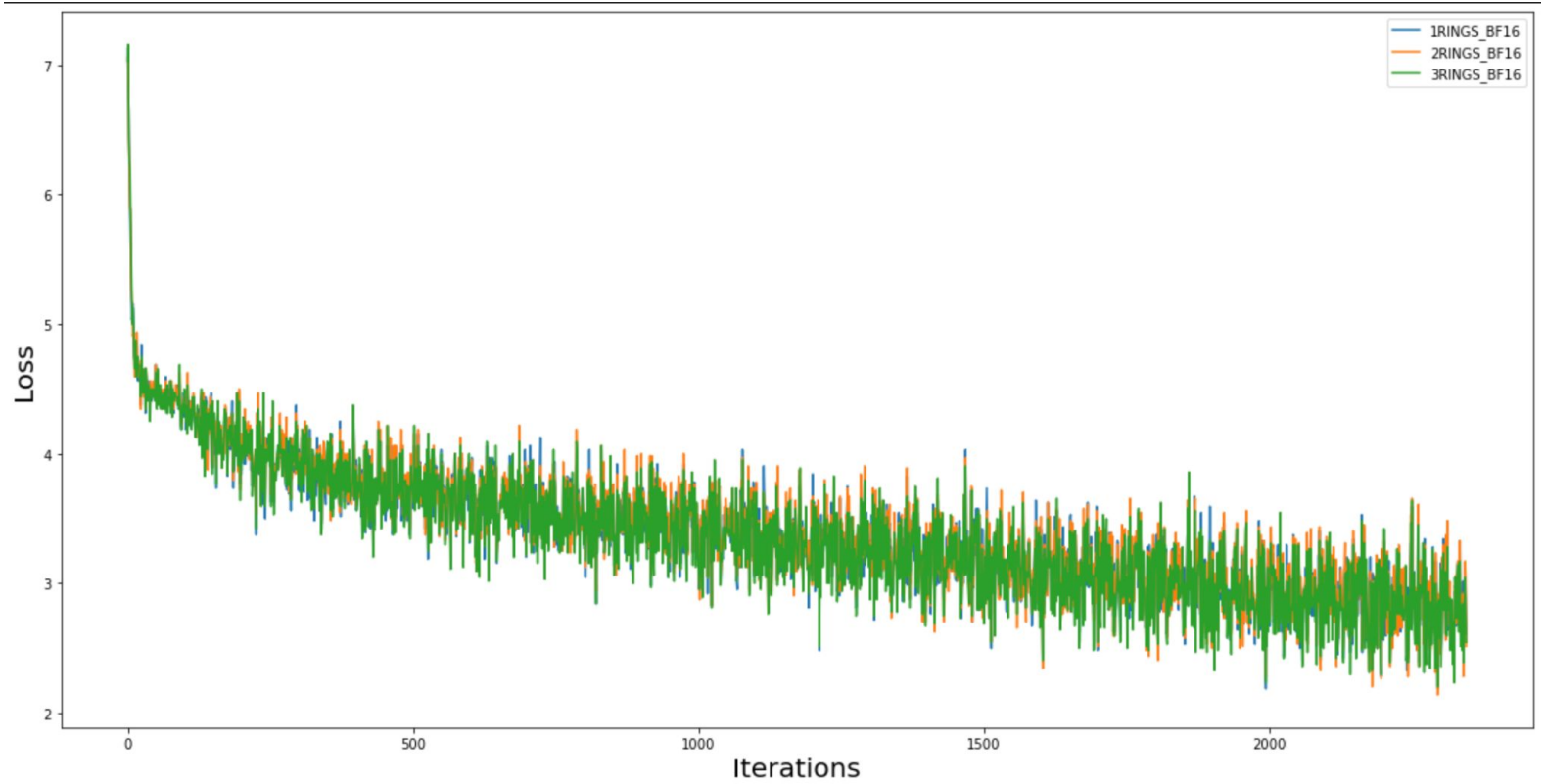
VGG16 -- bfloat16. SGD optimizer

I couldn’t get it to converge yet

DenseNet121 -- bfloat16. Adam optimizer

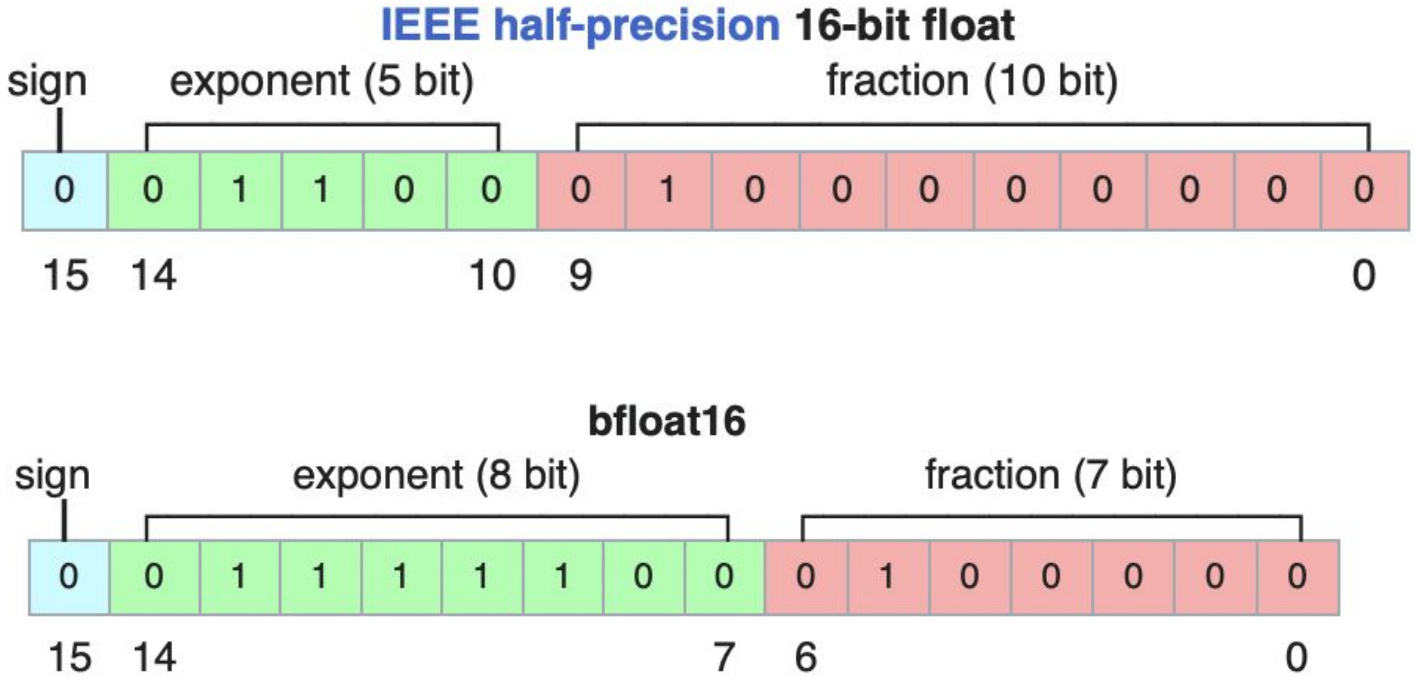


ResNet50 -- bfloat16. Adam optimizer



float16 and bfloat16 comparison

Reference


IEEE half-precision 16-bit float
sign
exponent (5 bit)
fraction (10 bit)
0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
15 14 10 9 0
bfloat16
sign
exponent (8 bit)
fraction (7 bit)
0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0
15 14 7 6 0

ResNet-50 -- float16. Accuracies on the train set

Rings	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	27.83%	78.59%	79.96%	91.97%	99.64%
2	28.05%	79.92%	82.21%	92.97%	99.68%
3	28.4%	79.74%	83.63%	92.85%	99.73%
4	28.98%	76.6%	83.67%	93.13%	99.65%

ResNet-50 -- float16. Accuracies on the train set.
Determinism disabled

Rings	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	%	%	%	%	%
2	%	%	%	%	%
3	%	%	%	%	%
4	%	%	%	%	%

ResNet-50 -- bfloat16. Accuracies on the train set

Rings	Acc. after Epoch 1	Acc. after Epoch 50	Acc. after Epoch 100	Acc. after Epoch 150	Acc. after Epoch 200
1	31.02%	63.91%	69.36%	76.82%	85.26%
2	28.17%	66.64%	68.57%	75.96%	85.14%
3	28.86%	64.24%	70.38%	70.15%	still running
4	still running	still running	still running	still running	still running