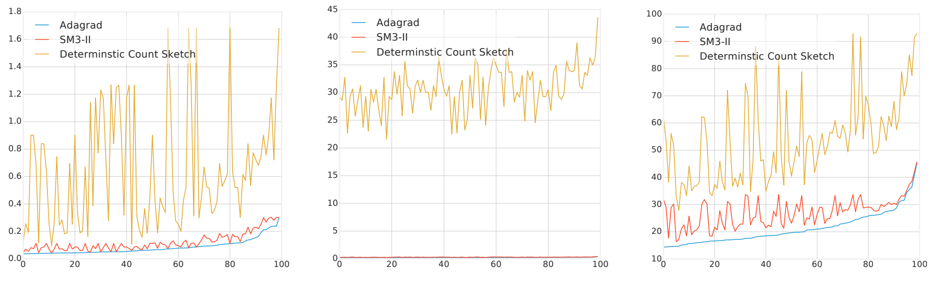
Reviewer 1: Thank you for the clear guidance on how to improve our presentation. We will follow it closely for the final version. That said, your review reflects much more enthusiasm than your score suggests. We would greatly appreciate if your could re-examine your numerical evaluation. • *Relation to AdaFactor:* Please see response to Reviewer 3. • *Open-sourcing:* Certainly! We have been finalizing an open-source version to be available soon on GitHub. • *Learning Rates (LR):* All algorithms we experimented with needed LR warm-up for improved performance. Beyond this, SM3 decays its LRs autonomously (like AdaGrad) and does *not* equire an "external" LR schedule. In contrast, Adam and AdaFactor do rely on an external LR schedules. Full details on the LRs and schedules used are provided in the supplementary. We will clarify this in the final version, thank you for pointing it out!

Reviewer 2: Thank you for the constructive comments. It appears that you are pleased by the high significance of the contributions, but have some concerns about novelty which we now address. • *Relation to "Compressing Gradient Optimizers" (SKMS'19):* First, we note that SKMS'19 should more fairly be seen as a concurrent work to ours rather than a prior work. (The first version of our paper appeared online before the first version of SKMS'19.) This note aside, SM3 is superior to the Count Sketch algorithm of SKMS'19 in several important ways:

(1) *Efficiency:* Randomized sketching is extremely inefficient on GPUs and TPUs as it requires sparse look-ups and is not cache-efficient. For this reason, SKMS'19 only apply their techniques to sparse (sampled) softmax and embedding layers with block versions of the hashes. They leave the treatment of hidden layers to future work. In contrast, our technique is deterministic and cache-friendly and "compresses" *all layers* of the model.
(2) *Space saving:* SKMS'19 uses sketches that are 5x smaller than the original tensors, whereas our sketches are **100x-1000x times smaller**.
(3) *Compression quality:* Empirically, the SM3 compression results in significantly smaller error compared to SKMS'19, as illustrated in the figure below. We plan to include these comparisons in the final version.
(4) *Empirical performance:* These differences allow us to show improvements on a large variety of tasks & models. SKMS'19 essentially brings no improvement for Imagenet, and only has improvements for models dominated by the embedding and softmax layers (along with sampled softmax). In contrast, SM3 converges faster for Imagenet, and for language models such as a state-of-the-art 24-layer BERT, we show significant improvements in convergence. We could **not** obtain reasonable results with SKMS'19 (and they do not report such results).

• *Applying to Adam:* SM3 can be used with exponential moving-averages (like Adam) instead of sums (like Adagrad) through a simple modification of $\nu'_t(i)$ (SM3-II): $\nu'_t(i) \leftarrow \beta \min_{r:S_r \ni i} \mu'_{t-1}(r) + (1-\beta) g_t^2(i)$. As we discuss in the paper, in our experiments moving-averages performed substantially worse than sums (regardless of memory savings).

Reviewer 3: • *LR schedules, compared to AdaFactor:* The discussion in Section 4 was meant to underscore that the LR in Adam/AdaFactor does not necessarily decay with time, and so practitioners often incorporate manually-tuned external decay schedules (see Table 4 in the supplementary). SM3 does *not* require any external decay schedule. That said, all algorithms we experimented with benefited from a short warmup phase of a few thousand updates. This is currently detailed in the supplementary material, and in the final version it will be made explicit in the main text. Thank you for pointing out the lack of clarity. • *AdaFactor for general tensors:* The AdaFactor paper did not describe an extension to higher order tensors, and indeed, the source code of AdaFactor breaks tensors into disjoint matrices (slices) each of which is approximated separately. Further, AdaFactor's approximations can be over or underestimates of the gradient moments, and in our experiments we see training instability which precluded the usage of AdaFactor with a batch size of 2048 (Figure 3 in the paper). SM3 does not suffer from this issue. Its *theoretical* and *empirical* convergence properties naturally extend to higher ranks. • *Breakdown of memory usage:* We will definitely try to elaborate more on this important point in the final version, space permitting. We would appreciate it if you could add some details in your final review about what kind of breakdown you would have liked to see. • *Analysis under different assumptions:* It would be indeed interesting to extend our analysis to various non-convex settings, and we are currently working in that direction. In terms of over-parameterization, is there a concrete assumption you think might improve convergence bounds? Please do add it to your final review, we would really appreciate it.



(a) Input embedding (b) Attention layer (c) Output softmax

Figure 1: Magnitude of the 100 largest accumulators of Adagrad and its approximation with SM3 and Count Sketch (SKMS'19) for a Transformer model trained on WMT'14 en→fr dataset.