Normalization Layer Per-Example Gradients are Sufficient to Predict Gradient Noise Scale in Transformers

Conference on Neural Information Processing Systems (NeurIPS)

Gavia Gray[†] , Aman Tiwari[♮], Shane Bergsma[†], Joel Hestness[†]

[†]Cerebras Systems, [♮]Subjective.dev

Choosing the Right Batch Size

Difficulties:

- As models get larger, larger batch sizes produce better results
- Larger batch sizes may be required to converge faster
- Testing different batch sizes with a grid search at large scale is not practical

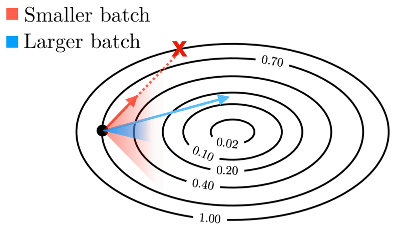
Choosing the Right Batch Size

Difficulties:

- As models get larger, larger batch sizes produce better results
- Larger batch sizes may be required to converge faster
- Testing different batch sizes with a grid search at large scale is not practical

GNS is a useful signal here.

Gradient Noise Scale



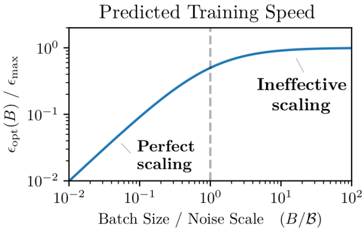


Figure: From McCandlish et al. (2018), illustrating the intuition of GNS and its usefulness in training.

Computing Gradient Noise Scale

$$G_{\mathrm{est}(\theta)} \sim \mathcal{N}\left(G(\theta), \frac{1}{B}\Sigma(\theta)\right) \quad \text{and} \quad \mathcal{B}_{\mathrm{simple}} = \frac{tr(\Sigma)}{G^T G}. \tag{1}$$

To compute this we need these estimators:

$$\|\mathcal{G}\|_2^2 := \frac{1}{B_{\mathrm{big}} - B_{\mathrm{small}}}\left(B_{\mathrm{big}}\left\|G_{B_{\mathrm{big}}}\right\|_2^2 - B_{\mathrm{small}}\left\|G_{B_{\mathrm{small}}}\right\|_2^2\right) \approx G^T G \tag{2}$$

$$\mathcal{S} := \frac{1}{1/B_{\mathrm{small}} - 1/B_{\mathrm{big}}}\left(\left\|G_{B_{\mathrm{small}}}\right\|_2^2 - \left\|G_{B_{\mathrm{big}}}\right\|_2^2\right) \approx tr(\Sigma), \tag{3}$$

$G_{B_{\mathrm{big}}}$ are normal minibatch gradients but $G_{B_{\mathrm{small}}}$ are microbatch gradients.

Reusing intermediate tensors

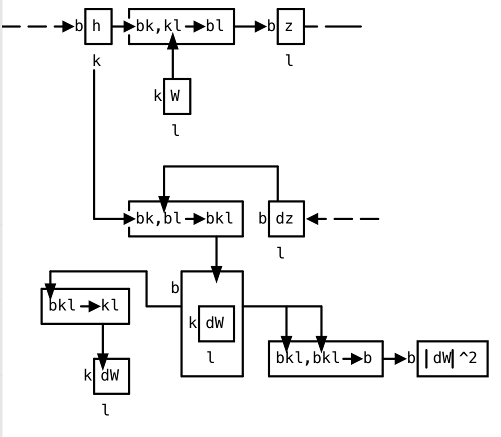
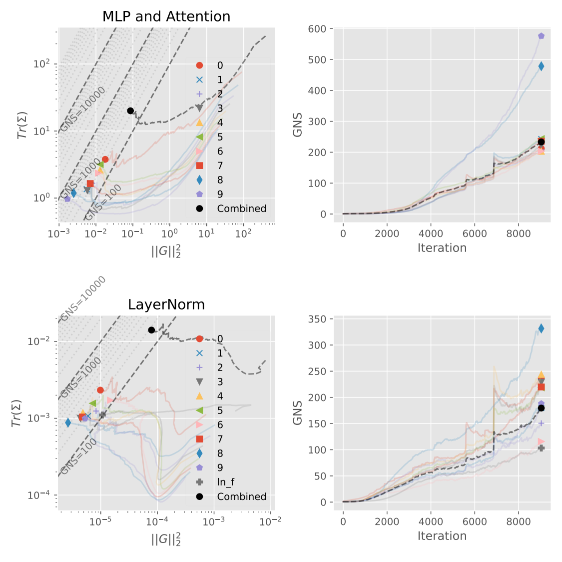


Figure: We can reuse the intermediate 3D per-example tensor.

GNS by Layer Type or Index



Efficient LayerNorm Kernel

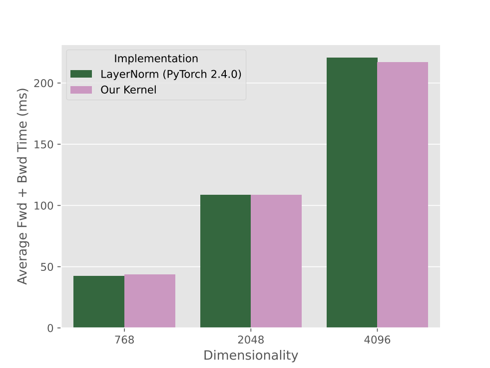


Figure: Speed comparison versus our fused custom kernel computing per-example gradient norms in tandem.

Universal GNS

Track GNS anywhere:

- Enable backward operations that extract per-example gradient norms while computing the parameter gradients

Universal GNS

Track GNS anywhere:

- Enable backward operations that extract per-example gradient norms while computing the parameter gradients
- Train at any scale on any number of devices

Universal GNS

Track GNS anywhere:

- Enable backward operations that extract per-example gradient norms while computing the parameter gradients
- Train at any scale on any number of devices
- MFU cost for exact tracking is 10-25% MFU in practice

Universal GNS

Track GNS anywhere:

- Enable backward operations that extract per-example gradient norms while computing the parameter gradients
- Train at any scale on any number of devices
- MFU cost for exact tracking is 10-25% MFU in practice
- Tracking norm layer GNS costs 0% MFU

Why You Should Look at GNS

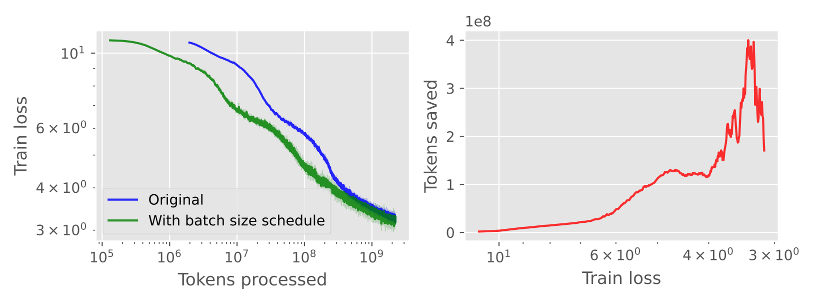


Figure: (Left) Linear batch size schedule tracking the GNS over 2.2 billion tokens processed. (Right) The number of tokens saved over the fixed batch size run to achieve the same loss.

Thanks

The code to replicate this work or use in future work may be found at: `https://github.com/CerebrasResearch/nanoGNS`.

References I

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training, 2018. URL https://arxiv.org/abs/1812.06162.