A GLUE Benchmark Details

The GLUE benchmark consists of 8 (originally 9) tasks [Wang et al., 2018]. Since there has been a Cambrian explosion of benchmarks since the halcyon days of GLUE, we elaborate on the individual GLUE benchmarks for reference:

A.1 Large Finetuning Datasets

MNLI (Multi-Genre Natural Language Inference) [392,702 train, 19,643 test] is a large crowd-sourced entailment classification task [Williams et al., 2017]. The model is given two sentences and has to predict whether the second sentence is entailed by, contradicts, or is neutral with respect to the first one. For example:

- Premise: "Buffet and a la carte available."
- Hypothesis: "It has a buffet."
- Label: 0 (entailment)

QNLI [104,743 train, 5,463 test] this Stanford Question Answering dataset consists of question-paragraph pairs drawn from Wikipedia [Rajpurkar et al., 2016].

QQP (Quora Question Pairs 2) [363,846 train, 390,965 test]. The task is to determine whether two sentences are semantically equivalent [Iyer et al., 2017].

A.2 Small Finetuning Datasets

RTE (Recognizing Textual Entailment) [2,490 train, 3,000 test] Given two sentences, the model has to predict whether the second sentence is or is not entailed by the first sentence [Dagan et al., 2006, Giampiccolo et al., 2007, Bentivogli et al., 2009]. Note that in our work we use a checkpoint from the MNLI finetuning to finetune on RTE.

CoLA (Corpus of Linguistic Acceptability) [8,551 train, 1,063 test] [Warstadt et al., 2019] is a benchmark with sentences that are either linguistically acceptable or grammatically incorrect. For example:

- "The higher the stakes, the lower his expectations are." Label: 1 (acceptable)
- "Mickey looked up it." Label: 0 (unacceptable)

SST-2 (Stanford Sentiment Treebank) [67,349 train, 1,821 test] consists of sentences from movie reviews. The task is to classify the sentiment as either positive or negative [Socher et al., 2013].

MRPC (Microsoft Research Paraphrase Corpus)[3,668 train, 1,725 test] [Dolan and Brockett, 2005] The dataset consists of sentence pairs extracted from online news sources. The task is to classify whether the sentences in the pair are semantically equivalent.

STSB (Semantic Textual Similarity Benchmark) [5,749 train, 1,379 test] This dataset contains sentence pairs that are given similarity scores from 0 to 5 [Cer et al., 2017].

Note that we excluded finetuning on the 9th GLUE task WNLI (Winograd NLI) [Levesque et al., 2012], as in the original BERT study (it is a very small dataset [634 train, 146 test] with a high number of adversarial examples). Finetuning on RTE, MRPC and STSB starts from a checkpoint already finetuned on MNLI (following the example of Izsak et al. [2021] and other studies). This is done because all the above tasks deal with sentence pairs, and this staged finetuning leads to consistent empirical improvement.

B Finetuning Hyperparameters

We used the hyperparameters in Table S1 for finetuning all BERT and RapidBERT models. All finetuning datasets used a max sequence length of 256 tokens. We found that these values worked well across all tasks for BERT-Base, RapidBERT-Base, and RapidBERT-Large; BERT-Large however was somewhat under-performant on QQP for some pretraining seeds.

Task	learning rate	beta	epsilon	weight decay	epochs
MNLI	5e-5	[0.9, 0.98]	1e-6	5e-6	3
QNLI	1e-5	[0.9, 0.98]	1e-6	1e-6	10
QQP	3e-5	[0.9,0.988]	1e-6	3e-6	5
RTE	1e-5	[0.9, 0.98]	1e-6	1e-5	3
CoLA	5e-5	[0.9, 0.98]	1e-6	5e-6	10
SST-2	3e-5	[0.9,0.988]	1e-6	3e-6	3
MRPC	8e-5	[0.9, 0.98]	1e-6	8e-6	10

Table S1: Finetuning hyperparameters for BERT and RapidBERT across Base and Large.

C RapidBERT-Large Multinode Throughput Scaling

The experiments in the main section of this paper were all performed on a single node with $8\times$ A100 GPUs. How well do our innovations to the BERT architecture maximize throughput at the multinode scale?

We measured the throughput of RapidBERT-Large (430M) during training on 8, 16, 32, 64, 128 and 200 GPUs, and plotted the tokens per second for various global batch sizes. Global batch size is an important factor in the throughput measurements; in general, cranking up the batch size increases the GPU utilization and raw throughput. As the number of nodes increases, the global batch size needs to be increased as well in order to maintain high throughput.

If the global batch size is kept constant while increasing the number of nodes, the throughput does not increase linearly. This can be seen in Figure S1; a global batch size of 4096 spread across 64 GPUs using Distributed Data Parallelism (DDP) means that each GPU will only apply matmul operations on matrices with a dimension of 64, which leads to suboptimal throughput. If the global batch size is increased to 65,536 across 64 GPUs, this roughly means that each GPU will apply matmul operations on matrices with a dimension of 1024, leading to higher throughput. However, such a large global batch size might not lead to the best downstream accuracy; this is a question that we were not able to address in this study due to resource and time constraints.

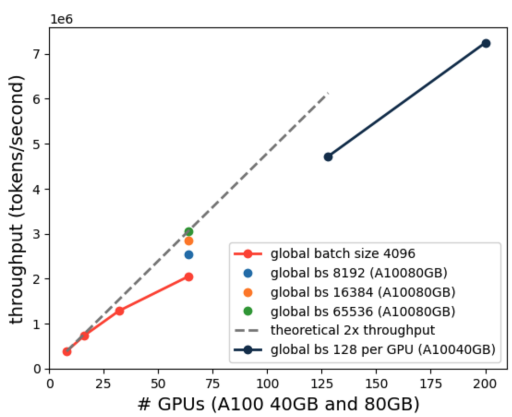

Figure S1: RapidBERT-Large (430M) multinode throughput scaling

D RapidBERT-Base Model FLOPs Utilization (MFU)

Model FLOPs Utilization (MFU) is an estimate of what percentage of the hardware's FLOPs are being used during training. The estimate is based on the measured throughput and the known FLOPs of the computation.

MFU calculates the utilization from the floating point operations required for a single forward/backwards pass of the model, and does not account for the additional compute required for other implementation details such as activation checkpointing. Thus, MFU is independent of

Model	Throughput (tokens /sec)	MFU	Hardware	Time to 79.6	Batch Size	Micro-batch Size
BERT Base	0.4e6	10.4%	8× A100 80	110.4 minutes (1.84 hours)	4096	512
RapidBERT Base	1.1e6	39.97%	8× A100 80	67.8 minutes (1.13 hours)	4096	512
RapidBERT Base	0.938e6	30.9%	8× A100 40	76.8 minutes	4096	128
RapidBERT Base	1.88e6	31.0%	16× A100 40	38.5 minutes	4096	128
RapidBERT Base	3.15e6	25.9%	32× A100 40	23.1 minutes	4096	128
RapidBERT Base	4.77e6	19.6%	64× A100 40	15.7 minutes	4096	64

Table S2: Multinode Throughput scaling for RapidBERT-Base

implementation and hardware. For more details, see Korthikanti et al. [2022]. All FLOP calculations exclude the operations required for normalization, activation, and residuals.

Following the notation in the PaLM paper [Chowdhery et al., 2022], Model FLOPs Utilization (MFU) is approximated as:

$$\text{MFU} = \frac{(6 \cdot n_{parameters} \cdot T_{observed})}{n_{gpus} \cdot T_{theoretical}} \tag{3}$$

where $T_{observed}$ is the observed throughput and $T_{theoretical}$ is the theoretical peak throughput.

In the numerator, the number of learnable parameters in the model is multiplied by a factor of 6 to estimate the matmul FLOPs per token seen ($2\times$ for the forward pass and $4\times$ for the backward pass). This is then multiplied by the number of tokens seen per second. As a first-order approximation, we exclude the extra FLOPs per token due to dense self-attention.

In the denominator, the theoretical peak throughput is provided in the GPU hardware specs. For A100 GPUs using `bfloat16`, this theoretical peak throughput is 312 teraFLOPs.

RapidBERT-Base Ave. GLUE Score	8×A100 80GB hours	8×A100 80GB cost ($2.50 GPU/hr)	8×A100 40GB hours	8×A100 40GB cost ($2 GPU/hr)
79.6	1.13	$22.60	1.28	$20.00
82.2	2.81	$56.20	3.19	$51.00
83.4	5.27	$105.40	5.99	$95.78

Table S3: RapidBERT-Base GLUE (dev) scores, time and cost comparison

E GPU Pricing

As of this writing, A100 GPU pricing ranges from $4.10 (40 GB) for on demand cloud compute on AWS, to $2.46 (40 GB) / $5.00 (80 GB) per GPU on GCP to $1.10 (40 GB) / $1.50 (80 GB) per GPU using Lambda labs. At an intermediate price of $2.50 an hour per A100 80 GB GPU, training to 79.6 GLUE average score takes 1.13 hours and costs roughly $22.60.[6] Some example costs are calculated in Table S3.

[6]See for example "Cloud GPU instances with the largest VRAM 2022" (https://medium.com/@aleixlopez/cloud-gpu-instances-to-solve-out-of-memory-error-2022-d5012883a272?)

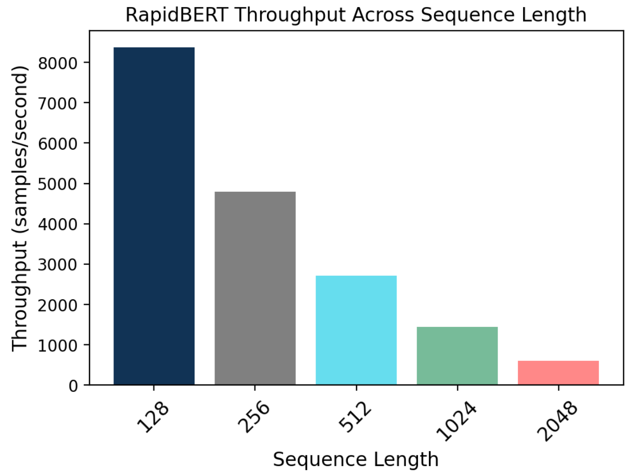

Figure S2: Throughput for Various Sequence Lengths

F Throughput as a Function of Sequence Length

In Figure S2, we plot the pretraining throughput of RapidBERT-Base with various context windows. As the sequence length doubles, the pretraining throughput halves. We note that for all of the pretraining in the main text, we use a maximum sequence length of 128.

G Gated Linear Units (GLU) Optimizations

GLU adds elementwise multiplication of two linear projections and shown to be quality improvements over standard Transformer block. There are multiple ways to implement GLUs and we experimented with a couple to pick the best performing one. Figure S3 shows standard feedforward transformer block (A) and two implementations of GLUs (B-C). "Fused GLU" in (C) fuses the two matrix multiplications into one and is expected to perform better in some domains.

Figure S4 shows the performance impact of the two GLU over standard feedforward transformer block (which would be 0% slowdown) for a single GPU. This figure only shows the performance of the forward pass, and backward is expected to behave similarly. We can draw two conclusions from this chart: 1) For smaller batch sizes, both GLU implementations add significant overhead over the standard block. 2) For batch sizes < 128, Fused GLU implementation is better than regular GLU implementation and beyond 128 it's slightly worse. The implementation used in the main text is the "Fused GLU" implementation (C) with batch size global 4096. Since the profiling in Figure S4 is per GPU, we are in the regime of $4096/8 = 512$.

The main reason for slowness of GLUs over standard block is extra elementwise multiplication in GLU layers. As for why fused implementation is slower, profiling analysis shows that the Linear layer ends up calling different CUDA kernels for matrix-multiplications and their relative performance is different for different sizes.

H Limitations and Broader Impact

H.1 Limitations

While we trained two different model sizes, we have not pretrained a RapidBERT model in the >1B parameter range. In this regime, it is possible there will be training stability issues; this is an area of future work.

We also only trained models for 70,000 steps and 178,000 steps of batch size 4096. It is possible that some of the Pareto properties change in the longer regime, although we suspect that this is unlikely.

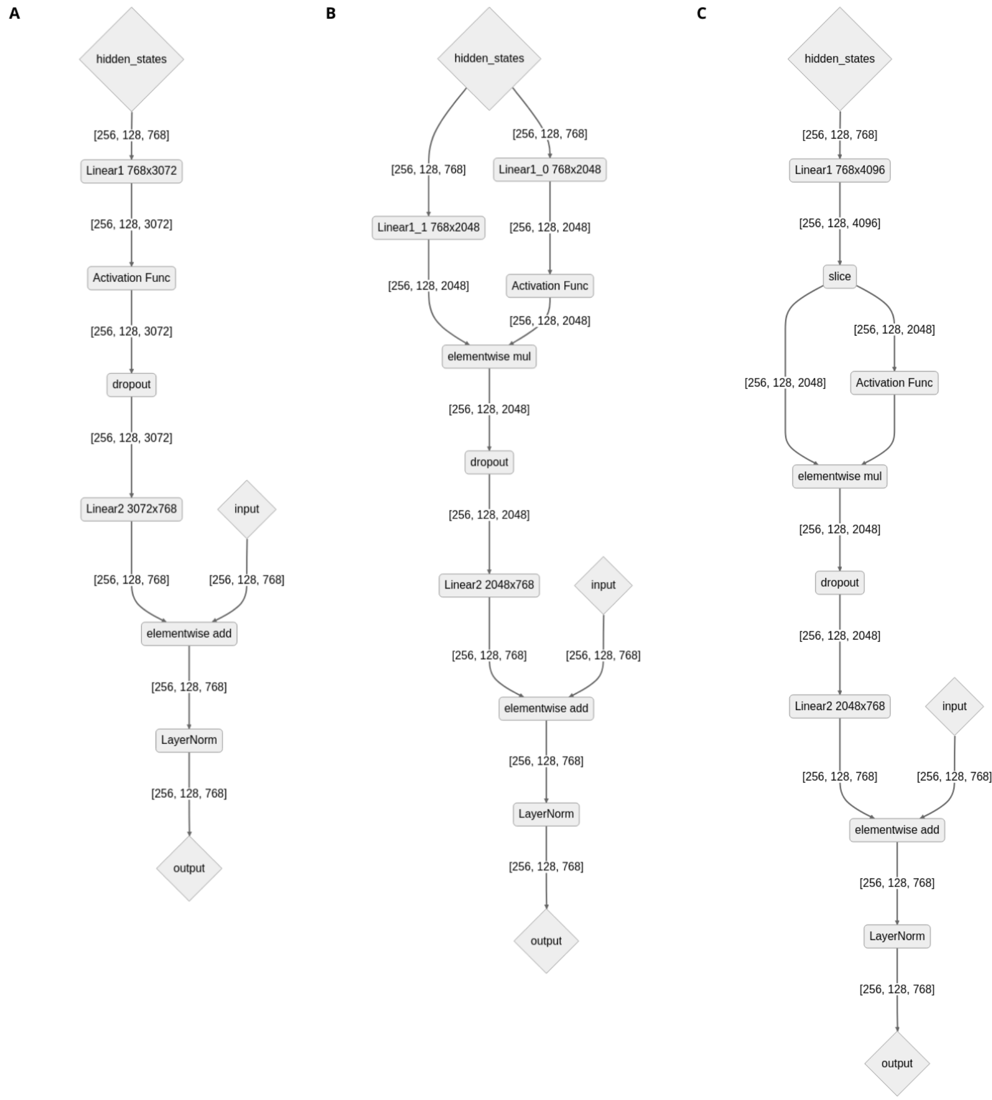

Figure S3: Standard FeedForward Transformer Block and Gated Linear Unit Modifications. Each edge shows the tensor dimensions and it's assuming a batch size of 256, sequence length of 128 and a hidden dim of 768. (A): A standard transformer feedforward block. (B): Naive implementation of a Gated Linear Unit. The number of parameters in this are the same as in (A). (C): Fused implementation of a Gated Linear Unit where the two matrix multiplications (`Linear1_0` and `Linear1_1`) from (B) are fused into one (`Linear1`) with 2× the parameters and the output is sliced. This is functionally equivalent to (B).

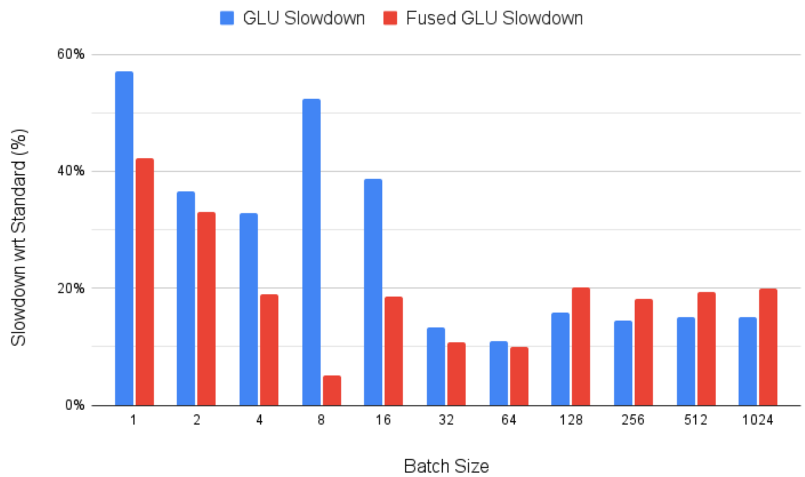

Figure S4: Slowdown of different implementations of Gated Linear Unit. This slowdown is with respect to standard feedforward transformer block. The number of parameters between standard feedforward transformer block and the two GLU implementations are the same.

541 ### H.2 Broader Impact

542 BERT models are highly used for NLP tasks. By open-sourcing this work, we hope that our code
543 and models will be used by the wider research community. We recognize however that models like
544 BERT and RapidBERT are tools that can be used for nefarious purposes, and that biases inherent in
545 the training data can be reflected in the final model artefacts.