Harder, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference

Anonymous ACL submission

Abstract

Encoder-only transformer models such as BERT offer a great performance-size tradeoff for retrieval and classification tasks with respect to larger decoder-only models. Despite being the workhorse of numerous production pipelines, there have been limited Pareto improvements to BERT since its release. In this paper, we introduce ModernBERT, bringing modern model optimizations to encoder-only models and representing a major Pareto improvement over older encoders. Trained on 2 trillion tokens with a native 8192 sequence length, ModernBERT models exhibit state-of-the-art results on a large pool of evaluations encompassing diverse classification tasks and both single and multi-vector retrieval on different domains (including code). In addition to strong downstream performance, ModernBERT is also the most speed and memory efficient encoder and is designed for inference on common GPUs.

1 Introduction

After the release of BERT (Devlin et al., 2019), encoder-only transformer-based (Vaswani et al., 2017) language models dominated most applications of modern Natural Language Processing (NLP). Despite the rising popularity of Large Language Models (LLMs) such as GPT (Radford et al., 2018, 2019; Brown et al., 2020), Llama (Touvron et al., 2023; Dubey et al., 2024), and Qwen (Bai et al., 2023; Yang et al., 2024), encoder-only models remain widely used in a variety of non-generative downstream applications.

The encoder's popularity is largely due to their modest inference requirements, enabling them to efficiently process corpora of documents at scale for retrieval and quickly perform discriminative tasks. Encoder models offer a compelling tradeoff in quality versus size, making them a popular option against encoder-decoder and decoder-only language models when dealing with substantial amounts of data (Penedo et al., 2024).

Encoder models are particularly popular in Information Retrieval (IR) applications, e.g., semantic search, with notable progress on leveraging encoders for this task (Karpukhin et al., 2020; Khattab and Zaharia, 2020). While LLMs have taken the spotlight in recent years, they have also motivated a renewed interest in encoder-only models for IR. Indeed, encoder-based semantic search is a core component of Retrieval-Augmented Generation (RAG) pipelines (Lewis et al., 2020), where encoder models are used to retrieve and feed LLMs with context relevant to user queries.

Encoder-only models are also still frequently used for a variety of discriminative tasks such as classification (Tunstall et al., 2022) or Natural Entity Recognition (NER) (Zaratiana et al., 2024), where they often match the performance of specialized LLMs. Here again, they can be used in conjunction with LLMs, for example detecting toxic prompts (Ji et al., 2023; Jiang et al., 2024b) and preventing responses, or routing queries in an agentic framework (Yao et al., 2023; Schick et al., 2023).

Surprisingly, these pipelines currently rely on older models, and quite often on the original BERT itself as their backbone (Wang et al., 2022; Xiao et al., 2023), without leveraging improvements developed in recent years. Practitioners face many drawbacks: sequence lengths limited to 512 tokens, suboptimal model design (Anthony et al., 2024) and vocabulary sizes (Karpathy, 2023), and generally inefficient architectures, whether in terms of downstream performance or computational efficiency. Finally, training data is limited in volume and restricted to narrow domains (especially lacking code data) or lacking knowledge of recent events.

Recent modernization efforts have only partially addressed the shortcomings of encoder-only models due to limited breadth. MosaicBERT (Portes et al., 2023), CrammingBERT (Geiping and Goldstein, 2023), and AcademicBERT (Izsak et al., 2021) focused on matching BERT performance with better training efficiency. NomicBERT (Nussbaum et al., 2024) and GTE-en-MLM (Zhang et al., 2024) (developed concurrently to this work) introduced longer-context encoder models focused on retrieval applications, but did not optimize for efficiency or classification performance, and re-used older training data mixtures which is especially apparent in programming-related tasks.

Contributions We present ModernBERT, a modernized encoder-only transformer model, with an improved architecture designed to increase downstream performance and efficiency, especially over longer sequence lengths. We also bring encoder-only models to modern, larger data scales, by training on 2 trillion tokens, with a data mixture including code data. We release two models, **ModernBERT-base** and **ModernBERT-large**, which reach state-of-the-art overall performance against all existing encoder models on a wide variety of downstream tasks. These results are achieved with considerably higher inference efficiency, processing sequences of 8192 tokens almost two times faster than previous models.

To support future research on encoder-only models, we release FlexBERT[1], our modular architecture framework allowing easy experimentation, and inspired by Pythia (Biderman et al., 2023), all intermediate training checkpoints (further detailed in Section 2.2.2).

2 Methods

2.1 Architectural Improvements

Our model architecture extends the standard transformer architecture (Vaswani et al., 2017) by incorporating extensively tested recent advances (Section 2.1.1). We introduce additional efficiency-oriented modifications, through both architectural and implementation improvements (Section 2.1.2) and a GPU optimized model design (Section 2.1.3). All of our architectural decisions were informed by ablations, which we detail in Appendix D.

2.1.1 Modern Transformer

Bias Terms Following (Dayma et al., 2021), we disable bias terms in all linear layers except for the final decoder linear layer[2]. We also disable all bias terms in Layer Norms (Xu et al., 2019). These two changes allow us to spend more of our parameter budget in linear layers.

Positional Embeddings We use rotary positional embeddings (RoPE) (Su et al., 2024) instead of absolute positional embeddings. This choice is motivated by the proven performance of RoPE in short- and long-context language models (Black et al., 2022; Dubey et al., 2024; Gemma et al., 2024), efficient implementations in most frameworks, and ease of context extension.

Normalization We use a pre-normalization block (Xiong et al., 2020) with the standard layer normalization (Lei Ba et al., 2016), which is known to help stabilize training (Xiong et al., 2020). Similar to CrammingBERT (Geiping and Goldstein, 2023) which also uses pre-normalization, we add a LayerNorm after the embedding layer. To avoid repetition, we remove the first layer norm in the first attention layer.

Activation We adopt GeGLU (Shazeer, 2020), a Gated-Linear Units (GLU)-based (Dauphin et al., 2017) activation function built on top of the original BERT's GeLU (Hendrycks and Gimpel, 2016) activation function. This is in line with recent work showing consistent empirical improvements when using GLU variants (Shazeer, 2020; Geiping and Goldstein, 2023).

2.1.2 Efficiency Improvements

Alternating Attention Following recent work on efficient long context models (Gemma et al., 2024), attention layers in ModernBERT alternate between global attention, where every token within a sequence attends to every other token, and local attention, where tokens only attend to each other within a small sliding window (Beltagy et al., 2020). In ModernBERT, every third layer employs global attention with a RoPE theta of 160,000 and the remaining layers use a 128 token, local sliding window attention with a RoPE theta of 10,000.

Unpadding ModernBERT follows MosaicBERT (Portes et al., 2023) and GTE (Zhang et al., 2024) in employing unpadding (Zeng et al., 2022) for both training and inference. Encoder-only language models typically use padding tokens to ensure a uniform sequence length in a batch,

[1]Link removed for review anonymity. FlexBERT is built on top of a revised MosaicBERT (Portes et al., 2023) codebase.

[2]While many efficient BERT training recipes disable the bias term in the decoder, e.g. Geiping and Goldstein (2023), we hypothesized a decoder bias might help alleviate weight tying's negative effects (Gao et al., 2019; Welch et al., 2020).

wasting compute on semantically empty tokens. Unpadding avoids this inefficiency by removing padding tokens, concatenating all sequences from a minibatch into a single sequence, and processing it as a batch of one. Prior unpadding implementations unpad and repad sequences internally for different model layers, wasting compute and memory bandwidth. We use Flash Attention's variable length attention and RoPE implementations, allowing jagged attention masks and RoPE applications on one unpadded sequence. ModernBERT unpads inputs before the token embedding layer and optionally repads model outputs leading to a 10-to-20 percent performance improvement over other unpadding methods.

Flash Attention Flash Attention (Dao et al., 2022) is a core component of modern transformer-based models, providing memory and compute efficient attention kernels. At the start of this work, Flash Attention 3 (Shah et al., 2024), the most recent iteration for Nvidia H100 GPUs, did not include support for sliding window attention. ModernBERT uses a mixture of Flash Attention 3 for global attention layers and Flash Attention 2 (Dao, 2023) for local attention layers.

torch.compile We leverage PyTorch's built-in compiling (Ansel et al., 2024) to improve the training efficiency by compiling all compatible modules. This yields a 10 percent improvement in throughput with negligible compilation overhead.

2.1.3 Model Design

At the same parameter count, models with more narrow layers (*Deep & Narrow*) have different learning patterns than models with fewer wide layers (*Shallow & Wide*) (Nguyen et al., 2021). Tay et al. (2022) and (Liu et al., 2024) have shown that *Deep & Narrow* language models have better downstream performance than their shallower counterparts, at the expense of slower inference.

Anthony et al. (2024) highlighted that large runtime gains can be unlocked by designing models in a *hardware-aware* way, which had previously been anecdotally observed by many practitioners (Shoeybi et al., 2019; Karpathy, 2023; Black et al., 2022). ModernBERT was designed through many small-scale ablations to maximize the utilization of a basket of common GPUs[3], while aiming to be as *Deep & Narrow* as possible without a significant inference slowdown.

[3]Which, at the time of this work, are server GPUs: NVIDIA T4, A10, L4, A100, and H100 and consumer GPUs: NVIDIA RTX 3090 and 4090. Prioritization was given to inference GPUs (excluding A100 & H100).

ModernBERT has 22 and 28 layers for the base and large models, for a total parameter count of 149 and 395 million, respectively, striking the balance between downstream performance and hardware efficiency. ModernBERT base has a hidden size of 768 with a GLU expansion of 2,304, while large has a hidden size of 1,024 and GLU expansion of 5,248. These ratios allow optimal tiling across tensor cores and the most efficient tiling across the differing number of streaming multiprocessors on our target basket of GPUs. More details on model design are provided in Appendix B.

2.2 Training

2.2.1 Data

Mixture Both ModernBERT models are trained on 2 trillion tokens of primarily English data from a variety of data sources, including web documents, code, and scientific literature, following common modern data mixtures. We choose the final data mixture based on a series of ablations.

Tokenizer Unlike the majority of recent encoders which reuse the original BERT tokenizer (Nussbaum et al., 2024; Portes et al., 2023; Zhang et al., 2024), we opt to use a modern BPE tokenizer. We use a modified version of the OLMo tokenizer (Groeneveld et al., 2024) which provides better token efficiency and performance on code-related tasks. The ModernBERT tokenizer uses the same special tokens (e.g., [CLS] and [SEP]) and templating as the original BERT model (Devlin et al., 2019), facilitating backwards compatibility. To ensure optimal GPU utilization (Anthony et al., 2024; Karpathy, 2023), the vocabulary is set to 50,368, a multiple of 64 and includes 83 unused tokens to support downstream applications.

Sequence Packing In order to avoid high minibatch-size variance within our training batches as a result of unpadding, we adopt sequence packing (Raffel et al., 2020; Krell et al., 2022) with a greedy algorithm, which resulted in a sequence packing efficiency of over 99 percent, ensuring batch size uniformity.

2.2.2 Training Settings

MLM We follow the Masked Language Modeling (MLM) setup used by MosaicBERT (Portes et al., 2023). We remove the Next-Sentence Prediction objective which introduces noticeable overhead for no performance improvement (Liu et al., 2019a;

Izsak et al., 2021), and use a masking rate of 30 percent, as the original rate of 15 percent has since been shown to be sub-optimal (Wettig et al., 2023).

Optimizer We use the StableAdamW optimizer (Wortsman et al., 2023), which improves upon AdamW (Loshchilov and Hutter, 2019) by adding Adafactor-style (Shazeer and Stern, 2018) update clipping as a per-parameter learning rate adjustment. StableAdamW's learning rate clipping outperformed standard gradient clipping on downstream tasks and led to more stable training. Hyperparameters details are given in Appendix A.

Learning Rate Schedule During pretraining, we use a modified trapezoidal Learning Rate (LR) schedule (Xing et al., 2018), also known as Warmup-Stable-Decay (WSD) (Zhai et al., 2022; Hu et al., 2024). After a short LR warmup, the trapezoidal schedule holds the LR constant for the majority of training, followed by a short LR decay. This schedule has been shown to match the performance of cosine scheduling (Hägele et al., 2024; Hallström et al., 2024) with the benefit of enabling continual training on any checkpoint without cold restart issues (Ash and Adams, 2019). Unlike most trapezoidal schedules, we use a $1 - \text{sqrt}$ LR decay (Hägele et al., 2024), as we found it to outperform linear and cosine decay.

We trained ModernBERT-base at a constant LR of 8e-4 for 1.7 trillion tokens following a 3 billion token warmup. After a 2 billion token warmup, we trained ModernBERT-large at a LR of 5e-4 for 900 billion tokens. We rolled back and restarted training at 5e-5 for the remaining 800 billion tokens after large's loss plateaued for a few hundred billion tokens at 5e-4.

Batch Size Schedule Batch size scheduling starts with smaller gradient accumulated batches, increasing over time to the full batch size. In ablations, this schedule accelerated training progress. We warmup the batch size from 768 to 4,608 over 50 billion tokens and from 448 to 4,928 over 10 billion tokens, for ModernBERT-base and -large, respectively, with an uneven token schedule so each batch size has the same number of update steps. Details are provided in Appendix A.1.

Weight Initialization and Tiling We initialize ModernBERT-base with random weights following the Megatron initialization (Shoeybi et al., 2019). For ModernBERT-large, we follow the Phi model family (Li et al., 2023; Javaheripi et al., 2023)[4] and initialize -large's weights from ModernBERT-base. In ablation runs, this consistently matched Phi's improved training results and greatly speed up the initial loss decrease of our model training[5]. Details are provided in Appendix A.2.

Context Length Extension After training on 1.7 trillion tokens at a 1024 sequence length and RoPE theta of 10,000, we extend the native context length of ModernBERT to 8192 tokens by increasing the global attention layer's RoPE theta to 160,000 and train for an additional 300 billion tokens. We first train at a constant lower learning rate[6] of 3e-4 for 250 billion tokens on an 8192 token mixture of the original pretraining dataset sampled following Fu et al. (2024). Next, we upsample higher-quality sources following Gao et al. (2024) and conduct the decay phase with a $1 - \text{sqrt}$ LR schedule over 50 billion tokens. This context extension process yielded the most balanced model on downstream tasks, as most of our ablations using only one of these strategies resulted in a performance loss on either retrieval or classification tasks.

3 Downstream Evaluation

We performed an extensive set of evaluations, across a large range of tasks, aiming to demonstrate the versatility of ModernBERT in common scenarios.

For all tasks, ModernBERT is evaluated against existing encoders of similar size. The BASE size, conventionally defined as under 150 million parameters, includes BERT-base (Devlin et al., 2019), DeBERTa-v3-base (He et al., 2023), RoBERTa-base (Liu et al., 2019a), as well as the more recent 8192 context NomicBERT (Nussbaum et al., 2024) and GTE-en-MLM-base (Zhang et al., 2024). The LARGE size, conventionally defined as above 300 million and under 500 million parameters, includes BERT-large-uncased (Devlin et al., 2019), DeBERTa-v3-large (He et al., 2023) and RoBERTa-large (Liu et al., 2019a) and GTE-en-MLM-large (Zhang et al., 2024).

3.1 Evaluation Setting

3.1.1 Natural Language Understanding

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is the standard Natural Language Understanding (NLU)

[4]As detailed in their 2023 NeurIPS presentation.

[5]This initialization reduced the amount of batch size and LR warmup needed for ModernBERT-large

[6]We only lowered the LR for ModernBERT-base, as large already decreased LR during the 1024 token training phase.

benchmark for encoder models, aiming to measure how well a model performs across a range of sentence or sentence-pair understanding tasks, such as sentiment detection (Liu et al., 2019b) or language entailment, through tasks such as MNLI (Williams et al., 2018). Although GLUE is often regarded as saturated by the best-performing models, such as large language models (Zhao et al., 2023), it remains one of the most commonly used evaluation suites for smaller encoder-based models, and provides a good impression of a model's performance on common classification tasks (Portes et al., 2023; Zhang et al., 2024; He et al., 2023).

We follow the practice of previous studies (Devlin et al., 2019; Liu et al., 2019a; He et al., 2023) and conduct a hyperparameter search on each GLUE subset (detailed in Appendix E.1) in order to provide values comparable to other models.[7]

3.1.2 Text Retrieval

Information Retrieval (IR) is one of the most common applications of encoder-only models,[8] where they are used to represent documents and queries in semantic search (Karpukhin et al., 2020). This domain has recently seen considerable growth and interest following the spread of LLMs where semantic search powered by lightweight models is used to provide relevant context to LLMs as part of Retrieval-Augmented Generation pipelines.

We evaluate models in both the single-vector Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) setting and the multi-vector ColBERT (Khattab and Zaharia, 2020) setting.

We report retrieval results on the popular BEIR evaluation suite (Thakur et al., 2021), the common standard for evaluating retrieval performance across a variety of tasks and domains, using the nDCG@10 metric. For each setting detailed below, we conduct a learning rate sweep based on results over a subset of the BEIR benchmarks to select the final model, detailed in Appendix E.2.

Single vector retrieval One of the most common approaches to neural retrieval using encoders is DPR (Karpukhin et al., 2020), where a single-vector is used to represent an entire document. The similarity between a query and a document can then be computed through distance operations, such as cosine similarity. Models are finetuned using contrastive learning to create representations which are close if a document is relevant to a query, and distant if not (van den Oord et al., 2018).

We train every base model using the MS-MARCO (Bajaj et al., 2016) dataset with mined hard negatives (Xuan et al., 2020) on 1.25M samples with a batch size of 16 and learning rate warmup for 5% of the training using sentence-transformers (Reimers and Gurevych, 2019).

Multi vector retrieval Multi-vector retrieval, championed by ColBERT (Khattab and Zaharia, 2020), seeks to mitigate lost information from compressing an entire sequence into a single vector. In multi-vector retrieval, each document is represented by all of its individual token vectors, and the similarity between a query and a document is computed using the MaxSim[9] operator.

We adopt the training setup of JaColBERTv2.5 (Clavié, 2024), an update on the ColBERTv2 (Santhanam et al., 2022) training procedure, with a batch size of 16 and a 5% learning rate warmup. We train all models by distilling the knowledge of a teacher model by using the KL-Divergence between the normalized teacher and student scores. Models are trained on 810k samples from MS-Marco (Bajaj et al., 2016) and teacher scores from BGE-M3 (Chen et al., 2024), using the PyLate library (Chaffin and Sourty, 2024).

3.1.3 Long-Context Text Retrieval

With a native 8192 context length, ModernBERT improves long-context performance over most existing encoders. However, there are relatively few standardized long-context benchmarks for encoder-only models, and most benchmarks, such as Needle-in-a-haystack (Kamradt, 2023) and RULER (Hsieh et al., 2024) are geared towards generative tasks. Given this limitation, we demonstrate improved long-context performance on the English subset of MLDR (Chen et al., 2024), a long-context retrieval benchmark comprised of over 200,000 long documents. We evaluate three settings:

Single Vector – Out-Of-Domain Models are trained on short-context MS-MARCO as described above, and is evaluated on long context MLDR without any further fine-tuning.

[7]As (Zhang et al., 2024) do not explicitly mention a parameter sweep, we initially ran the same hyperparameter sweep as we did for ModernBERT, but observed inconsistencies in the results. To avoid under-representing GTE-en-MLM's capabilities, we choose to use their reported GLUE results.

[8]At the time of this paper's writing, over half of the 100 most downloaded models on the HuggingFace Model Hub were encoder-based retrieval models.

[9]The sum for every query token of its similarity with the most similar document token

	Model	IR (DPR) BEIR	IR (DPR) $MLDR_{OOD}$	IR (DPR) $MLDR_{ID}$	IR (ColBERT) BEIR	IR (ColBERT) $MLDR_{OOD}$	NLU GLUE	Code CSN	Code SQA
Base	BERT	38.9	23.9	32.2	49.0	28.1	84.7	41.2	59.5
Base	RoBERTa	37.7	22.9	32.8	48.7	28.2	86.4	44.3	59.6
Base	DeBERTaV3	20.2	5.4	13.4	47.1	21.9	88.1	17.5	18.6
Base	NomicBERT	41.0	26.7	30.3	49.9	61.3	84.0	41.6	61.4
Base	GTE-en-MLM	41.4	**34.3**	**44.4**	48.2	69.3	85.6	44.9	71.4
Base	ModernBERT	**41.6**	27.4	44.0	**51.3**	**80.2**	**88.4**	**56.4**	**73.6**
Large	BERT	38.9	23.3	31.7	49.5	28.5	85.2	41.6	60.8
Large	RoBERTa	41.4	22.6	36.1	49.8	28.8	88.9	47.3	68.1
Large	DeBERTaV3	25.6	7.1	19.2	46.7	23.0	**91.4**	21.2	19.7
Large	GTE-en-MLM	42.5	**36.4**	**48.9**	50.7	71.3	87.6	40.5	66.9
Large	ModernBERT	**44.0**	34.3	48.6	**52.4**	**80.4**	90.4	**59.5**	**83.9**

Table 1: Results for all models across an overview of all tasks. CSN refers to CodeSearchNet and SQA to StackQA. $MLDR_{ID}$ refers to in-domain (fine-tuned on the training set) evaluation, and $MLDR_{OOD}$ to out-of-domain.

Single Vector – In Domain Models trained on MS-MARCO are further fine-tuned on long-context MLDR training set before being evaluated.

Multi-Vector – Out-Of-Domain Due to its token-level MaxSim mechanism, ColBERT models are able to generalize to long-context without any specific training (Bergum, 2024). We directly evaluate the best checkpoints from Section 3.1.2 without any further fine-tuning on MLDR.

3.1.4 Code Retrieval

Fueled by increasingly good code completion models (Jiang et al., 2024a), downstream applications have quickly grown in popularity following the emergence of code assistants.[10] Encoder-only models are used to process and retrieve large quantities of code-related information under resource constraints, increasing the importance of measuring and improving code capabilities of encoder models (Li et al., 2024). Unlike most previous encoders which were largely trained only on textual data (Devlin et al., 2019; Liu et al., 2019a; Portes et al., 2023; Zhang et al., 2024; Nussbaum et al., 2024), ModernBERT is pre-trained on code and uses a code-aware tokenizer[11].

To measure programming-related performance, we evaluate all models on CodeSearchNet (Husain et al., 2019), a code-to-text benchmark where the model must identify relevant docstring or comments for code blocks, and StackOverflow-QA (Li et al., 2024), where the model must identify relevant responses to StackOverflow questions, in a "hybrid" setting where documents contain both text and code. The latter benchmark also leverages long-context capabilities, as its queries and documents respectively contain 1,400 and 1,200 words on average, leading to average token counts of over 2000.

We evaluate these benchmarks using the CoIR (CodeIR) framework (Li et al., 2024), as single-vector retrieval tasks. All models are trained by re-using the best hyper-parameters identified in Section 3.1.2.

3.2 Downstream Results and Discussion

Aggregated results for all evaluations are presented in Table 1. For BEIR and GLUE, the two common evaluation suites, we follow existing practice in reporting the average results. Detailed results are provided in Appendix E.

In terms of downstream performance, ModernBERT is the strongest overall model at both the BASE and LARGE model sizes. ModernBERT represents a Pareto improvement on all tasks over the original BERT and RoBERTA models, with better performance on every evaluation category.

Short-Context Retrieval On BEIR, both variants of ModernBERT outperform existing encoders in both the DPR and ColBERT settings, including the recent GTE-en-MLM and NomicBERT models designed to serve as better backbones for retrieval (Zhang et al., 2024; Nussbaum et al., 2024).

While ModernBERT-base only narrowly edges out GTE-en-MLM-base on DPR evaluations,

[10]Spearheaded by GitHub Copilot in 2021

[11]Avoiding issues such as the ones seen in T5 (Raffel et al., 2020), whose vocabulary did not include curly braces.

	Model	Params	Short BS	Short Fixed	Short Variable	Long BS	Long Fixed	Long Variable
Base	BERT	110M	1096	**180.4**	90.2	–	–	–
	RoBERTa	125M	664	179.9	89.9	–	–	–
	DeBERTaV3	183M	236	70.2	35.1	–	–	–
	NomicBERT	137M	588	117.1	58.5	36	46.1	23.1
	GTE-en-MLM	137M	640	123.7	61.8	38	46.8	23.4
	GTE-en-MLM$_{\text{xformers}}$	137M	640	122.5	128.6	38	47.5	67.3
	ModernBERT	149M	**1604**	148.1	**147.3**	**98**	**123.7**	**133.8**
Large	BERT	330M	**792**	**54.4**	27.2	–	–	–
	RoBERTa	355M	460	42.0	21.0	–	–	–
	DeBERTaV3	434M	134	24.6	12.3	–	–	–
	GTE-en-MLM	435M	472	38.7	19.3	28	16.2	8.1
	GTE-en-MLM$_{\text{xformers}}$	435M	472	38.5	40.4	28	16.5	22.8
	ModernBERT	395M	770	52.3	**52.9**	**48**	**46.8**	**49.8**

Table 2: Memory (max batch size, *BS*) and Inference (in thousands of tokens per second) efficiency results on an NVIDIA RTX 4090, averaged over 10 runs. Dashes indicate unsupported configurations.

ModernBERT-large increases its lead despite having comparatively fewer parameters at 395M to GTE-en-MLM-large's 435M.

Long-Context Retrieval - Single Vector In the DPR setting, ModernBERT achieves impressive performance on MLDR, a long-context text retrieval task. However, these results also highlight an interesting phenomenon: without long-context finetuning ModernBERT outperforms both shorter-context models and the long-context NomicBERT but performs noticeably worse than GTE-en-MLM. The performance gap narrows considerably when evaluated in-domain, with both models performing similarly. This suggests that ModernBERT can effectively process long context sequences as a dense encoder but may require more adapted tuning. We plan to explore multiple potential explanations for this phenomenon in future work, including the impact of local attention or GTE-en-MLM having spent a larger part of its pretraining compute budget on longer sequence lengths (Zhang et al., 2024).

Long-Context Retrieval - Multi-Vector In the ColBERT setting, long-context models (GTE-en-MLM, NomicBERT, and ModernBERT) all outperform short-context models by at least 40 NDCG@10 points without requiring any specific finetuning. These results confirm the findings of Bergum (2024), who showed that ColBERT models are particularly well-suited to long-context retrieval tasks. Among the long-context models, ModernBERT outperforms other long-context models, with at least a 9 NDCG@10 point lead on both model sizes. We theorize that these sizable gains could be explained by our long pretraining ensuring few, if any, tokens are under-trained, as well as a potentially synergistic effect of local attention with ColBERT-style retrieval, but leave further exploration of this phenomenon to future work.

Natural Language Understanding Both ModernBERT models demonstrate exceptional NLU results, as measured by GLUE. ModernBERT-base surpasses all existing base models, including DeBERTaV3-base, becoming the first MLM-trained model to do so. This is surprising, as DeBERTaV3 was trained with the Replaced-Token-Detection objective, which was previously thought to yield stronger downstream NLU performance (Clark et al., 2020; He et al., 2023). ModernBERT-large is the second-best large encoder on GLUE, almost matching DeBERTaV3-large with one-tenth fewer parameters while processing tokens in half the time (see Section 4).

Code On programming tasks, in both code-to-text (CodeSearchNet) and longer-context hybrid settings (StackQA), ModernBERT outperforms all other models. This result was expected, as it is the only evaluated encoder to be trained on a data mixture including programming data. These results, combined with ModernBERT's strong showings on other tasks, indicates that ModernBERT has improved understanding of code at no detriment to its ability to process natural text.

4 Efficiency

4.1 Evaluation Setting

To measure inference efficiency across multiple sequence lengths, we create 4 synthetic sets of 8192 documents[12]. The first two document sets are fixed-length: in *fixed short-context*, all documents contain 512 tokens and in *fixed long-context* all documents contain 8192 tokens[13]. To account for the impact of unpadding, we also create two varying-length document sets, where the number of tokens in each set are defined by a normal distribution centered on half the maximum sequence length, 256 and 4096 tokens, respectively. Full data statistics are provided in Appendix F.

We then evaluate all models based on the number of tokens they can process per second, averaged over ten runs. All efficiency evaluations are ran on a single NVIDIA RTX 4090, one of the target GPUs of ModernBERT outlined in Section 2.1.3 We evaluate the GTE-en-MLM models under two settings: out-of-the box, and with the use of the xformers (Lefaudeux et al., 2022) library, which enables efficiency enhancements such as unpadding.

4.2 Results

All tokens-per-second efficiency results are presented in Table 2, with absolute run-times provided in Appendix F. ModernBERT stands out as the most efficient model overall. On short context, it processes fixed-length 512 token inputs faster than all other recent encoders, although slower than the original BERT and RoBERTa models[14]. On long-context, ModernBERT is faster than all competing encoders, processing documents 2.65 and 3 times faster than the next-fastest encoder at the BASE and LARGE sizes, respectively. ModernBERT-large's processing speed at length 8192 (46,801 tokens per second) is closer to that of GTE-en-MLM base (47,507 tokens per second) than it is to GTE-en-MLM-large (16,532 tokens per second).

On variable-length inputs, both GTE-en-MLM and ModernBERT models are considerably faster than all other models, largely due to unpadding. However, ModernBERT remains noticeably more efficient than GTE-en-MLM, processing 14.5-30.9 percent more tokens per second at low context lengths and 98.8-118.8 percent more at longer context lengths, thanks to its use of local attention.

ModernBERT is the overall most memory efficient model on both model sizes. ModernBERT-base is able to process batch sizes twice as large as every other model on both input lengths. ModernBERT-large is slightly less memory efficient than the original BERT-large on short-context inputs, but can process batches at least 60 percent bigger than every other large model.

5 Conclusion

We present ModernBERT, an open family of encoder-only models which set a new state of the art over existing encoder models on a wide range of classification and retrieval tasks. We show that encoders benefit from both recent pretraining data scales and architecture improvements from autoregressive LLMs.

ModernBERT has a native sequence length of 8,192 tokens and incorporates recent architecture improvements, such as GeGLU layers, RoPE positional embeddings, and alternating local-global attention. ModernBERT is the first open model to feature entire model unpadding and is the first encoder designed in a hardware-aware way to maximize inference efficiency.

ModernBERT pushes the encoder state of the art forward across a wide range of benchmarks. On GLUE, ModernBERT-base is the first encoder to beat DeBERTaV3-base since its release in 2021. ModernBERT is in a class of its own in ColBERT-style code and long-context retrieval benchmarks, scoring at least 6.85 and 9.1 percentage points higher than the closest model, respectively, while remaining state-of-the-art on short-context retrieval in both single and multi-vector settings.

At the same time, ModernBERT processes short context inputs twice as fast as DeBERTaV3 and long-context inputs two times faster than the next fastest model with best-in-class memory efficiency.

ModernBERT is a generational leap over the original encoder models, with notable performance improvements over BERT and RoBERTa on both classification and retrieval tasks. ModernBERT is one of the few encoders to support long-context and programming applications, while simultaneously setting a new record in encoder inference efficiency.

[12]Many common benchmarks are biased towards low and uniform sequence lengths, which is unrepresentative of many real-world situations.

[13]512 being the maximum length of most existing encoders, while 8192 is the maximum length of all long-context ones.

[14]This is partially due to the relatively low parameter count of BERT and RoBERTa compared to more recent encoders.

6 Limitations

Language This study focuses exclusively on the English language, and trains on a very large number of tokens. As such, a major limitation of our work is that it is not directly applicable to other languages, and potentially even less-so to lower resources languages.

Biases Our model is trained largely on web data, as a result, all of its representations are subject to the biases present in such data.

Harmful Content Generation The MLM objective gives the model some ability to generate text by suggesting a given token to replace the [MASK] token (Samuel, 2024), which could result in the generation of harmful content. However, ModernBERT is not, primarily, a generative model, and as such, has not been trained to and therefore cannot generate longer sequences of text. As a result, it is considerably less likely to be at risk of generating harmful content of any kind.

MLM-only objective Given the strong results of DeBERTav3 on classification tasks but weak ones on retrieval, it seems that a training leveraging both MLM and RTD might be better suited to achieve best results on classification. Extending our work to RTD is thus a promising line of research.

Scaling Besides the architectural modifications, a key aspect of our studies is data scaling. However, other scaling axes, notably in terms of model parameters are left unexplored.

References

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, volume 2, pages 929–947.

Quentin Anthony, Jacob Hatef, Deepak Narayanan, Stella Biderman, Stas Bekman, Junqi Yin, Aamir Shafi, Hari Subramoni, and Dhabaleswar Panda. 2024. The case for co-designing model architectures with hardware. *Preprint*, arXiv:2401.14489.

Jordan T. Ash and Ryan P. Adams. 2019. On the difficulty of warm-starting neural network training. *CoRR*, abs/1910.08475.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Jo Kristian Bergum. 2024. Announcing vespa long-context ColBERT. *Vespa Blog*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Antoine Chaffin and Raphaël Sourty. 2024. Pylate: Flexible training and retrieval for late interaction models.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 2318–2335. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Benjamin Clavié. 2024. Jacolbertv2.5: Optimising multi-vector retrievers to create state-of-the-art japanese retrievers with constrained resources. *Preprint*, arXiv:2407.20750.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phúc Lê Khắc, Luke Melas, and Ritobrata Ghosh. 2021. Dall·e mini.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *Preprint*, arXiv:2402.10171.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models. *ArXiv*, abs/1907.12009.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2024. How to train long-context language models (effectively). *Preprint*, arXiv:2410.02660.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single GPU in one day. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 11117–11143. PMLR.

Team Gemma, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro von Werra, and Martin Jaggi. 2024. Scaling laws and compute-optimal training beyond fixed training durations. *CoRR*, abs/2405.18392.

Oskar Hallström, Said Taghadouini, Clément Thiriet, and Antoine Chaffin. 2024. Passing the torch: Training a mamba model for smooth handover.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zhen Leng Thai, Kai Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *CoRR*, abs/2404.06395.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling laws and compute-optimal training beyond fixed training durations. *Preprint*, arXiv:2405.18392.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10644–10652, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *arXiv preprint arXiv:2307.04657*.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024a. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. 2024b. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Preprint*, arXiv:2406.18510.

Gregory Kamradt. 2023. Needle In A Haystack - pressure testing LLMs. *Github*.

Andrej Karpathy. 2023. The most dramatic optimization to nanogpt so far (25% speedup) is to simply increase vocab size from 50257 to 50304 (nearest multiple of 64).

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.

Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. 2022. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *Preprint*, arXiv:2107.02027.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. 2022. xformers: A modular and hackable transformer modelling library. `https://github.com/facebookresearch/xformers`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *ArXiv e-prints*, pages arXiv–1607.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9459–9474.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Yichun Yin, Hao Zhang, Yong Liu, Yasheng Wang, and Ruiming Tang. 2024. Coir: A comprehensive benchmark for code information retrieval models. *arXiv preprint arXiv:2407.02883*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *Preprint*, arXiv:2309.05463.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *Preprint*, arXiv:2402.14905.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. 2021. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *CoRR*, abs/2402.01613.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Preprint*, arXiv:2406.17557.

Jacob Portes, Alexander Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2023. Mosaicbert: A bidirectional encoder optimized for fast pretraining. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Rushi Qiang, Ruiyi Zhang, and Pengtao Xie. 2024. Bilora: A bi-level optimization framework for overfitting-resilient low-rank adaptation of large pre-trained models. *CoRR*, abs/2403.13037.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskeve. 2018. Improving language understanding by generative pre-training. In *OpenAI Tech Report*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. Scaling language models: Methods, analysis & insights from training gopher. *Preprint*, arXiv:2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

David Samuel. 2024. Berts are generative in-context learners. *CoRR*, abs/2406.04823.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3715–3734. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings* *of Machine Learning Research*, pages 4596–4604. PMLR.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2022. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations (ICLR) 22*.

The Mosaic ML Team. 2021. composer. `https://github.com/mosaicml/composer/`.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *arXiv preprint*.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, volume 54, pages 1–12. ACM New York, NY, USA.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Benjamin Warner. 2023. optimī: Fast, modern, memory efficient, and low precision pytorch optimizers.

Charles Welch, Rada Mihalcea, and Jonathan K. Kummerfeld. 2020. Improving low compute language modeling with in-domain embedding initialisation. *Preprint*, arXiv:2009.14109.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2023. Should you mask 15% in masked language modeling? *Preprint*, arXiv:2202.08005.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. 2023. Stable and low-precision training for large-scale vision-language models. *Preprint*, arXiv:2304.13013.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. 2018. A walk with sgd. *Preprint*, arXiv:1802.08770.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32.

Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2020. Hard negative examples are hard, but useful. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 126–142. Springer.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. Gliner: Generalist model for named entity recognition using bidirectional transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5364–5376.

Jinle Zeng, Min Li, Zhihua Wu, Jiaqi Liu, Yuang Liu, Dianhai Yu, and Yanjun Ma. 2022. Boosting distributed training performance of the unpadded bert model. *arXiv preprint arXiv:2208.08124*.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 1204–1213. IEEE.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 1393–1412. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Training Settings

Detailed training settings can be found in Table 3.

During training we used MNLI as a live evaluation, along with validation loss and token accuracy metrics on a 500 million randomly sampled sequences from the source datasets.

We use Composer (Team, 2021) as our training framework and optimī (Warner, 2023) for our optimizer implementations.

A.1 Batch Size Schedule

Batch size warmup is a common-knowledge trick to speed up model training when working with medium to large batch sizes. Instead of "wasting" a full batch on updating the suboptimal initial weight distribution, we update the model weights on a gradually increasing batch size. Batch size warmup is usually longer than learning rate warmup, and can be thought of as providing a higher initial learning rate with a mini learning rate decay to the defined learning rate schedule. We warmup ModernBERT's batch size from 768 to 4,608 over 50 billion tokens and from 448 to 4,928 over 10 billion tokens, for -base and -large, respectively, with an uneven token schedule so each batch size has the same number of update steps.

A.2 Weight Tiling

Following the Phi family of models (Li et al., 2023; Javaheripi et al., 2023), we initialized ModernBERT-large directly from ModernBERT-base's pretraining weights using center tiling and Gopher layer scaling (Rae et al., 2022). Since Base's weight matrices are smaller than Large's, we centered Base' weights, accounting for each token embedding and attention head, then filled rest the of the weights using wraparound. Like Phi, we tested center initialization with random edge values and tiling from an edge, but both of these underperformed center tiling with wraparound. This weight initialization strategy greatly accelerates ModernBERT-large's initial training.

A.3 Weight Decay

We did not apply weight decay to the bias terms or normalization layers. Instead of PyTorch-style decoupled weight decay, we applied fully decoupled weight decay following Loshchilov and Hutter (2019).

A.4 Final Checkpoints

Inspired by recent work showing that checkpoint averaging yields stronger final models (Dubey et al., 2024; Clavié, 2024), we selected our final checkpoints by experimenting with various averaging methods and evaluating them on a subset of evaluation tasks. In no cases did Exponential Moving Average during annealing, as used by Dubey et al. (2024), result in stronger performance. ModernBERT-base is the result of averaging the 3 best performing annealing checkpoints with the final one. Averaging did not yield successful results on the large size, ModernBERT-Large model is the best performing annealing checkpoint.

	Pretraining Phase		Context Extension: Phase One		Context Extension: Phase Two	
	Base	Large	Base	Large	Base	Large
Training Tokens	1.719 trillion		250 billion		50 billion	
Max Sequence Length	1,024		8,192		8,192	
Batch Size	4,608	4,928	72	77	72	78
Warmup (tokens)	50 billion	10 billion	-	-	-	-
Microbatch Size	96	56	12	7	12	6
Learning Rate	8e-4	5e-4, 5e-5	3e-4	5e-5	3e-4	5e-5
Schedule	Trapezoidal		-	-	1-sqrt	
Warmup (tokens)	3 billion	2 billion	-	-	-	-
Decay (tokens)	-	-	-	-	50 billion	
Weight Decay	1e-5	1e-5, 1e-6	1e-5	1e-6	1e-5	1e-6
Total Time (hours)	194.2	425.3	39.9	80.7	11.5	21.7
Training Time (hours)	191.1	420.4	36.3	75.1	7.5	15.3
Model Initialization	Megatron	From Base	-	-	-	-
Dropout (attn out)	0.1					
Dropout (all other layers)	0.0					
Optimizer	StableAdamW					
Betas	(0.90, 0.98)					
Epsilon	1e-06					
Training Hardware	8x H100					
Training Strategy	Distributed DataParallel					
Software Libraries	PyTorch 2.4.0, Cuda 12.4.0, Composer 0.24.1, Flash Attention 2.6.3, FA3 commit 32792d3					

Table 3: ModernBERT training settings. Dropout and below are shared across all phases.

B Model Design

From Anthony et al. (2024), in addition to setting attention heads as multiples of 64 and setting the embedding matrix as a power of 2 or multiple of 64, there are three model design choices to maximize performance (assuming float16 or bfloat16 computation):

- **Tensor Core Requirement**: Weight matrix dimensions should be divisible by 64
- **Tile Quantization**: Weight matrix is divisible into 128 × 256 blocks.
- **Wave Quantization**: Number of blocks is divisible by the number of streaming multiprocessors (SM).

Given that we wanted to target good performance across multiple GPUs with a wide variety of SM counts, wave quantization is an impossible ask. So we selected a basket of GPUs (NVIDIA T4, A10, L4, RTX 3090, RTX 4090, A100, and H100) and calculated the approximate SM utilization for each by dividing the modulus blocks by the number of SMs. This appeared to be a decent performance heuristic in our spot checking. We then designed our models to maximize performance on the basket of GPUs, putting more weight on inference GPUs.

C Training Log

C.1 Sampling Issue

Our first pretraining run of ModernBERT-base ended in disaster as the loss exhibited a seesaw pattern before slowly diverging. Despite using PyTorch's distributed random sampler, training metrics suggested that the model was training on the dataset in a non-random order. Like the Olmo authors[15], we determined that the PyTorch random sampler returns sequentially biased samples when the number of samples is somewhere between 500 million and 1 billion samples[16]. We resolved this issue by replacing the PyTorch sampler with NumPy's PCG64DXSM random sampler.

C.2 Large Rollback

We rolled back and restarted ModernBERT-large training at a lower learning rate of 5e-5 and lower weight decay of 1e-6 for the last 800 billion tokens. Prior to restarting training, large's training loss, validation metrics, and live evaluations on MNLI had plateaued for a few hundred billion tokens at the higher 5e-4 learning rate. In contrast,

[15]We found a comment and GitHub issue about this in the Olmo codebase after resolving the issue ourselves.

[16]We did not conduct a rigorous statistical analysis to determine exactly when this happens.

	Base	Large
Vocabulary	50,368	50,368
Unused Tokens	83	83
Layers	22	28
Hidden Size	768	1024
Transformer Block	Pre-Norm	Pre-Norm
Activation Function	GeLU	GeLU
Linear Bias	False	False
Attention	Multi-head	Multi-head
Attention Heads	12	16
Global Attention	Every three layers	Every three layers
Local Attention Window	128	128
Intermediate Size	1,152	2,624
GLU Expansion	2,304	5,248
Normalization	LayerNorm	LayerNorm
Norm Epsilon	1e-5	1e-5
Norm Bias	False	False
RoPE theta	160,000	160,000
Local Attn RoPE theta	10,000	10,000

Table 4: ModernBERT model design

ModernBERT-base showed a continuous, but diminishing, improvement on training loss, validation metrics, and live evaluations through the entire 1.719 trillion token training phase. This highlights one of the risks of training with a constant learning rate, other learning rate schedules can mitigate selecting a too high learning rate (or too small batch size) by lowering the learning rate throughout training.

D Architecture ablations

To select the updates to add in the ModernBERT architecture, we performed different ablations, except where stated, most ablations where ran at the 8-20 billion token scale:

- We compared two GLU layers, GeGLU and SwiGLU. We find close to no difference between the two and choose to use GeGLU layers.

- Using different percentage of the head dimension for the RoPE dimension (50, 75, 100). Lower percentages gave slightly better results. However, the observed difference was minimal. As the ablations were conducted at a considerably smaller scale than the final training, we choose to err on the side of caution and opt to keep the dimension at 100 % to avoid potentially hindering the capabilities of the fully trained models.

- Both LayerNorm and RMSNorm yielded very similar results. While RMSNorm is theoretically faster, at the time this work was conducted, PyTorch did not have a native RMSNorm implementation, leading to eager-mode RMSNorm being the default implementation used for many users. To ensure ModernBERT has the highest possible out-of-the-box efficiency, we choose to use LayerNorm in the final models.

- We investigated using parallel attention to compute the MLP and attention matrices at the same time, which has been shown to increase processing speeds for larger model sizes (Chowdhery et al., 2023). However, for models within our targe sizes and pre-training sequence length, the speed-up we observed was minimal while we encountered significant degradation in downstream performance. As such, we do not use parallel attention. It is however possible that larger encoders and/or larger sequence lengths might see a different trade-off.

- We explored the use of alternating global/local attention, with global attention every 3 layers and local attention over a 128 token sliding window otherwise. This setup yielded identical downstream performance when compared to the use of global attention in every layer, even at 100 billion tokens, while resulting in major speedups.

- We experimented with multiple tokenizers, before selecting our final one, based on a modified OLMo (Groeneveld et al., 2024) tokenizer, which performed the best out of the recent tokenizers evaluated. Tokenizers from the BERT and RoBERTa generation of encoder models had competitive downstream performance on MNLI, but we theorized that their lack of recent training data and lack of code support would hinder downstream applications. Interestingly, we observed significant downstream performance degradation when using the Llama 2 (Touvron et al., 2023) tokenizer.

E Extended results

E.1 Full GLUE results

The results for all the models each GLUE subsets are presented in Table 5. The values for prior models are extracted from the literature. As mentioned in Section 3.1.1, we follow standard practice (Liu et al., 2019a; Portes et al., 2023; He et al., 2023) and conduct an hyperparameter search on each subset. More specifically, we perform a sweep over learning rates in $[1e{-}5, 3e{-}5, 5e{-}5, 8e{-}5]$, weight decay in $[1e{-}6, 5e{-}6, 8e{-}6, 1e{-}5]$, and number of epochs in $[1, 2, 3]$ for tasks in SST-2, MNLI, and RTE, and $[2, 5, 10]$ for tasks in QNLI, QQP, CoLA, MRPC, and STS-B. The final values are detailed in Table 6. Early stopping is used for all the fine-tuning runs which reduces the overall fine-tuning time considerably. RTE MRPC and STS-B checkpoints are trained starting from the MNLI checkpoint.

E.2 Full BEIR results

In the main body, we only report the average score over the 15 very diverse datasets of BEIR. We report the results on every subsets for both single and multi-vector retrieval in Table 7 and Table 8 respectively. For both settings and for every model, we perform a sweep for learning rates in $[1e{-}5, 2e{-}5, 3e{-}5, 5e{-}5, 8e{-}5, 1e{-}4]$ and choose the model obtaining the best average result over a subset of datasets composed of NFCorpus, SciFact, TREC-Covid and FiQA as the final model. Best learning rates for every setting are reported in Table 9. Although ModernBERT showcase strong results across the board, it should be noted that an important factor in its performance is TREC-COVID (Voorhees et al., 2021), potentially showcasing the benefits of ModernBERT being trained with a more recent knowledge cutoff than most existing encoders. However, NomicBERT and GTE have also been trained on updated data, so the cutoff cannot be the only factor affecting the performance.

F Efficiency

Full statistics of the synthetic datasets used to evaluate the efficiency of the models in Section 4 are given in Table 10. The detailed runtimes, alongside with the maximum batch size for every model is detailed in Table 11.

The high maximum batch-size achieved by ModernBERT models, considerably higher than any other models, highlight the strong memory efficiency of the model at both sizes. Inversely, it is worth noting that while DeBERTaV3 has competitive GLUE performance, it stands out as particularly inefficient, both in its memory use and processing speed. Indeed, on both model sizes, DeBERTaV3's memory use is 5-to-7 times higher than ModernBERT's, and it processes inputs, even in the most favorable scenario where all sequences are at the maximum possible length, thus negating any advantage from unpadding.

G Licensing

We release the ModernBERT model architectures, model weights, training codebase under the Apache 2.0 license. We omit links to these resources in this submission in the name of review anonymity.

					Single Sentence		Paraphrase and Similarity			Natural Language Inference		
Model	Params	Pos.	Seq.	Avg.	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
Base												
BERT$^{\beta}$	110M	Abs.	512	84.7	59.0	93.1	89.5	89.4	91.4	85.4	91.6	78.2
RoBERTa$^{\alpha}$	125M	Abs.	512	86.4	63.6	94.8	90.2	91.2	91.9	87.6	92.8	78.7
DeBERTav3$^{\epsilon}$	183M	Rel.	512	88.09	**69.19**	95.63	89.46	91.60	**92.4**	**90.01**	**94.03**	83.75
MosaicBERT-128$^{\beta}$	137M	Alibi	128	85.4	58.2	93.5	89.0	90.3	92.0	85.6	91.4	83.0
NomicBERT-2048$^{\gamma}$	137M	RoPE	2048	84	50	93	88	90	92	86	92	82
GTE-en-MLM$^{\delta}$	137M	RoPE	8192	85.61	57.02	93.35	92.14	90.21	88.78	86.69	91.85	84.84
ModernBERT	149M	RoPE	8192	**88.44**	65.14	**95.99**	**92.16**	**91.77**	92.11	89.06	93.94	**87.36**
Large												
BERT$^{\beta}$	330M	Abs.	512	85.2	56.2	93.3	87.8	90.6	90.9	86.3	92.8	83.8
RoBERTa$^{\alpha}$	355M	Abs.	512	88.9	68.0	96.4	90.9	92.4	92.2	90.2	94.7	86.6
DeBERTav3$^{\zeta}$	434M	Rel.	512	**91.37**	**75.3**	96.9	92.2	**93.0**	**93.3**	**91.8**	**96.0**	**92.7**
GTE-en-MLM$^{\delta}$	434M	RoPE	8192	87.58	60.39	95.07	**93.45**	91.37	89.19	89.20	93.90	88.09
ModernBERT	395M	RoPE	8192	90.46	71.39	**97.13**	91.67	92.78	92.66	90.83	95.20	92.06

Table 5: GLUE (Wang et al., 2018) dev set scores. $^{\alpha}$ taken from Table 8 of (Liu et al., 2019a), $^{\beta}$ taken from Table S3 of (Portes et al., 2023), $^{\gamma}$ from Table 2 of (Nussbaum et al., 2024), $^{\delta}$ from Table 21 of (Zhang et al., 2024), $^{\epsilon}$ from Table 2 of (Qiang et al., 2024) and $^{\zeta}$ from Table 3 of (He et al., 2023)

	Base			**Large**		
Task	LR	WD	Ep	LR	WD	Ep
CoLA	$8e{-}5$	$1e{-}6$	5	$3e{-}5$	$8e{-}6$	5
MNLI	$5e{-}5$	$5e{-}6$	1	$3e{-}5$	$1e{-}5$	1
MRPC	$5e{-}5$	$5e{-}6$	10	$8e{-}5$	$5e{-}6$	2
QNLI	$8e{-}5$	$5e{-}6$	2	$3e{-}5$	$5e{-}6$	2
QQP	$5e{-}5$	$5e{-}6$	10	$5e{-}5$	$8e{-}6$	2
RTE	$5e{-}5$	$1e{-}5$	3	$5e{-}5$	$8e{-}6$	3
SST-2	$8e{-}5$	$1e{-}5$	2	$1e{-}5$	$1e{-}6$	3
STSB	$8e{-}5$	$5e{-}6$	10	$8e{-}5$	$1e{-}5$	10

Table 6: Fine-tuning hyperparameters for ModernBERT on GLUE tasks. LR: Learning Rate, WD: Weight Decay, Ep: Epochs.

Model	Avg.	NFCorpus	SciFact	TREC-Covid	FiQA	ArguAna	Climate-FEVER	DBPedia	FEVER	HotpotQA	MSMARCO	NQ	Quora	SciDocs	Touche2020	CQADupstack
Base																
BERT	38.86	24.25	51.31	49.52	22.78	31.64	21.86	28.19	64.13	47.89	58.53	37.94	83.11	12.89	20.4	28.46
RoBERTa	37.74	20.36	45.61	52.22	26.1	35.18	22.29	23.12	60.2	44.95	55.99	34.74	84.03	11.44	**21.07**	28.81
DeBERTav3	20.21	8.04	22.64	48.42	11.52	26.05	9.74	5.33	17.29	7.96	25.24	12.51	74.69	5.36	14.2	14.21
NomicBERT	40.95	25.65	51.98	62.97	23.54	35.5	22.86	**30.33**	64.95	48.03	60.6	**42.55**	84.49	12.63	19.04	29.15
GTE-en-MLM	41.43	**26.26**	54.1	49.68	**30.11**	**35.71**	**24.51**	28.91	**66.53**	**49.93**	**63.05**	41.74	85.21	**14.13**	19.09	32.49
ModernBERT	**41.60**	23.71	**56.97**	**72.07**	28.75	35.69	23.55	23.8	59.94	46.06	61.6	39.48	**85.89**	12.51	20.83	**33.12**
Large																
BERT	38.88	23.33	50.67	48.86	23.98	35.23	22.08	**27.18**	61.73	45.91	59.8	39.47	83.64	12.98	19.47	28.91
RoBERTa	41.44	23.88	53.42	54.98	33.39	37.55	**23.47**	25.4	65.22	47.08	60.36	43.33	85.82	13.65	21.12	32.96
DeBERTav3	25.6	9.63	31.22	56.59	15.77	26.29	14.44	6.82	29.37	15.27	32.37	21.47	79.11	7.04	18.77	19.91
GTE-en-MLM	42.46	**27.72**	57.63	48.35	**34.01**	35.25	23.99	27	**65.36**	**50.8**	64.06	44.9	85.33	**15.63**	21.42	35.5
ModernBERT	**43.99**	26.21	**60.37**	**74.12**	33.13	**38.21**	20.51	25.1	62.68	49.21	**64.93**	**45.5**	**86.54**	13.81	**23.07**	**36.47**

Table 7: BEIR (Thakur et al., 2021) nDCG@10 scores for single-vector retrieval models.

Model	Avg.	NFCorpus	SciFact	TREC-Covid	FiQA	ArguAna	Climate-FEVER	DBPedia	FEVER	HotpotQA	MSMARCO	NQ	Quora	SciDocs	Touche2020	CQADupstack
Base																
BERT	49.04	34.18	71.48	69.88	35.01	**49.91**	19.18	42.36	83.10	69.76	45.42	55.38	84.08	14.66	26.97	34.24
RoBERTa	48.70	33.70	70.80	69.76	37.37	48.90	18.91	39.28	81.20	66.08	43.73	56.26	83.58	14.82	31.70	34.43
DeBERTav3	47.11	31.94	68.50	75.47	35.46	46.47	18.30	35.56	78.14	65.34	39.45	50.43	83.67	14.57	31.08	32.34
NomicBERT	49.89	**35.46**	72.18	73.45	35.85	44.84	19.03	**43.55**	83.88	**71.11**	**46.25**	**58.46**	84.00	15.12	31.30	33.88
GTE-en-MLM	48.21	35.07	71.51	69.42	35.97	48.48	17.41	41.15	79.85	67.01	44.37	52.82	85.19	15.01	25.35	34.59
ModernBERT	**51.31**	35.22	**72.95**	**80.47**	**37.99**	49.13	**22.19**	41.95	**85.76**	70.44	45.38	57.08	**86.25**	**16.01**	**33.85**	**35.12**
Large																
BERT	49.54	34.61	72.85	68.76	35.49	48.30	19.65	42.39	83.64	70.70	45.94	57.15	84.76	15.16	28.89	34.85
RoBERTa	49.76	34.96	72.32	74.36	38.72	50.04	19.55	40.97	82.00	66.22	44.72	57.47	85.92	15.26	27.90	**36.04**
DeBERTav3	46.73	31.69	70.20	73.34	34.99	46.18	18.04	36.49	78.96	63.20	39.42	51.64	81.08	14.08	28.55	33.09
GTE-en-MLM	50.65	35.16	72.43	67.21	39.55	50.25	20.79	**44.43**	82.50	72.00	**46.95**	**60.05**	**86.42**	15.87	30.89	35.38
ModernBERT	**52.35**	**36.04**	**73.17**	**81.32**	**40.34**	**50.34**	**22.29**	44.13	**85.80**	**72.52**	45.95	59.89	86.06	**16.90**	**34.59**	35.94

Table 8: BEIR (Thakur et al., 2021) nDCG@10 scores for multi-vector retrieval models.

Model	Single-vector (DPR)	Multi-vector (ColBERT)
Base		
BERT	$5e{-}5$	$8e{-}5$
RoBERTa	$3e{-}5$	$8e{-}5$
DeBERTav3	$8e{-}5$	$5e{-}5$
NomicBERT	$5e{-}5$	$1e{-}4$
GTE-en-MLM	$5e{-}5$	$8e{-}5$
ModernBERT	$8e{-}5$	$1e{-}4$
Large		
BERT	$3e{-}5$	$1e{-}4$
RoBERTa	$3e{-}5$	$1e{-}5$
DeBERTav3	$8e{-}5$	$1e{-}5$
GTE-en-MLM	$3e{-}5$	$3e{-}5$
ModernBERT	$1e{-}4$	$3e{-}5$

Table 9: Learning rate used for reported results on BEIR (Thakur et al., 2021) for both single and multi vector retrieval

	Fixed Short	Variable Short	Fixed Long	Variable Long
Total Token Count	4,194,304	2,096,510	67,108,864	33,604,913
Standard deviation	0	64	0	1024
Average Length	512	256	8192	4102
Longest sequence token count	512	476	8192	7624
Shortest sequence token count	512	32	8192	171
Number of sequences	8192	8192	8192	8192

Table 10: Token statistics for the synthetic datasets used in efficiency evaluations.

	Param Count	**Short Context (<=512 tokens)**			**Long Context (<=8192 tokens)**		
		Max bsize	**Runtime (in seconds)**		**Max bsize**	**Runtime (in seconds)**	
Base			Fixed	Variable		Fixed	Variable
BERT	110M	1096	23.25 ± 0.02		N/A	Unsupported	
RoBERTa	125M	664	23.32 ± 0.19		N/A	Unsupported	
DeBERTaV3	183M	236	59.71 ± 0.11		N/A	Unsupported	
NomicBERT	137M	588	35.83 ± 0.01		36	1455.46 ± 0.31	
GTE-en-MLM$_{\text{xformers}}$	137M	640	34.23 ± 0.10	16.30 ± 0.04	38	1412.60 ± 3.19	499.22 ± 0.11
GTE-en-MLM	137M	640	33.91 ± 1.21		38	1434.69 ± 3.69	
ModernBERT	149M	**1604**	28.33 ± 0.55	**14.23 ± 0.01**	**98**	**542.38 ± 0.20**	**251.18 ± 0.32**
Large							
BERT	330M	**792**	77.10 ± 1.50		N/A	Unsupported	
RoBERTa	355M	460	99.79 ± 1.79		N/A	Unsupported	
DeBERTaV3	434M	134	170.79 ± 0.06		N/A	Unsupported	
GTE-en-MLM$_{\text{xformers}}$	435M	472	108.98 ± 0.14	51.86 ± 0.02	28	4059.14 ± 4.55	1476.31 ± 0.94
GTE-en-MLM	435M	472	108.39 ± 0.07		28	4144.65 ± 0.05	
ModernBERT	395M	770	80.14 ± 1.65	**39.63 ± 0.02**	**48**	**1433.89 ± 0.99**	**674.88 ± 0.15**

Table 11: Inference runtime for all models.