Training compute-optimal transformer encoder models

Megi Dervishi [1,2] **Alexandre Allauzen** [2,3] **Gabriel Synnaeve** [1] **Yann LeCun** [1,4]

[1] FAIR at Meta, [2] Miles Team, LAMSADE, Université Paris-Dauphine, PSL, Paris, France,
[3] ESPCI PSL, Paris, France, [4] New York University

Correspondence: megidervishi@meta.com

Abstract

Transformer encoders are critical for a wide range of Natural Language Processing (NLP) tasks, yet their compute–efficiency remains poorly understood. We present the first comprehensive empirical investigation of compute-optimal pretraining for encoder transformers using the Masked Language Modeling (MLM) objective. Across hundreds of carefully controlled runs we vary model size, data size, batch size, learning rate, and masking ratio, with increasing compute budget. The compute-optimal data-to-model ratio of Transformer encoder models is 10 to 100 times larger than the ratio of auto-regressive models. Using these recipes, we train **OptiBERT**, a family of compute-optimal BERT-style models that matches or surpasses leading baselines—including ModernBERT and NeoBERT—on GLUE and MTEB while training with dramatically less FLOPS.

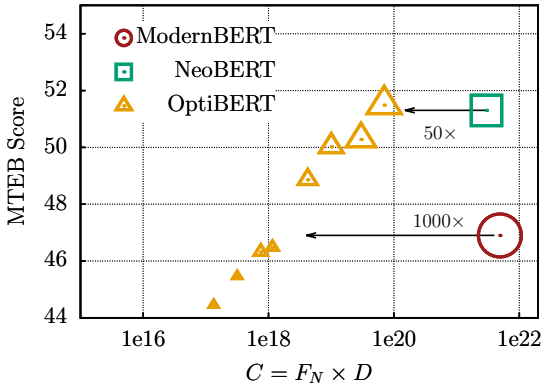


Figure 1: **The performance on MTEB** of current SOTA models NeoBERT$_{4096}$ (Breton et al., 2025) and ModernBERT$_{large}$(Warner et al., 2024) taken from Breton et al. 2025 and our compute-optimal BERT family (OptiBERT) in orange triangles as a function of the compute C measured in FLOPS.

1 Introduction

In the last few years, considerable effort has been devoted to improve decoder-only (Vaswani et al., 2023; Radford et al., 2018; Grattafiori et al., 2024; Groeneveld et al., 2024; DeepSeek-AI et al., 2024) Large Language Models (LLMs) trained to perform Next Token Prediction (NTP). In contrast, encoder-only models (Devlin et al., 2019; Liu et al., 2019a) trained with the Masked Language Modeling (MLM) objective have received comparatively less attention, despite being the backbone of a wide range of applications, including retrieval augmented generation (Lewis et al., 2020; Ram et al., 2023), toxicity detection (Hartvigsen et al., 2022; Ji et al., 2023; Jiang et al., 2024) and recommendation systems (Karpukhin et al., 2020; Khattab and Zaharia, 2020). The encoder literature (Li et al., 2023; Morris and Rush, 2024; Sturua et al., 2024) has primarily focused on fine-tuning BERT based models (Devlin et al., 2019; Liu et al., 2019a; He et al., 2020; Conneau et al., 2019) to improve the quality of their embeddings. However, these models rely on aging backbones, with BERT and RoBERTa now over five years old. Recent work has shifted towards updating the pretraining of the underlying architectures (Breton et al., 2025; Warner et al., 2024), incorporating modern techniques inspired by decoder-only models (Grattafiori et al., 2024; DeepSeek-AI et al., 2024). However, these efforts rely on ad hoc choices, lacking a systematic understanding of their scaling behavior.

Such an understanding has been central to the progress of decoder-based models, where scaling behavior is well characterized: larger models trained on more data consistently achieve lower training loss and better downstream performance. To avoid costly exhaustive empirical sweeps, researchers have developed *scaling laws*, which express performance as a function of model and dataset size. These laws are especially useful

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 35614–35629
November 4-9, 2025

for identifying *compute-optimal* configurations. A compute-optimal setting corresponds to the best trade-off between model size and dataset size to maximize performance under a fixed compute budget. Most studies (Kaplan et al., 2020; Hoffmann et al., 2022; DeepSeek-AI et al., 2024) have focused on describing such relations for auto-regressive LLMs trained with NTP, which is very different from the MLM of BERT-based models. A notable exception is the study by Urbizu et al. 2023a, however they focus on the low-resource regime and their analysis only considers three model scales, restricting a realistic extrapolation to more general trends and larger scales.

In this work, we conduct a comprehensive investigation of scaling laws for encoder-only Transformer models trained with MLM. First we examine the relation of learning rate and batch size (Fig. 2). Then we study the optimal scaling strategy for data and model on a fixed compute budget (Fig. 3 and 4). We discover that **encoder-only models have a more 'data-hungry' behavior compared to decoder-only models** and that the current decoder-only laws do not apply to encoders (see Fig. 5). In Section 4.3, we further show that this data hungriness does not depend on the masking ratio of the MLM objective (Clark et al., 2020).

Based on such laws, we confirm that the current state-of-the-art (SOTA) BERT-like models (Warner et al., 2024; Breton et al., 2025) are sub-optimal and over-trained. As shown in Fig. 1, we are able to train a family of compute-optimal BERT (OptiBERT) that **achieve/surpass the performance of SOTA with 50 to 1000 times less compute**. Our aim is to lay the groundwork for future compute-optimal scaling of backbone BERT models.

2 Method

Inspired by studies done for decoder-only models (DeepSeek-AI et al., 2024; Kaplan et al., 2020; Hoffmann et al., 2022) we aim to find the optimal scaling relation between data and model size under a certain compute budget.

Data. All models are trained on the `Fineweb_edu` dataset. This corpus is English only and contains over 1 trillion tokens, which is significantly larger than our longest training ($\sim$ 350B tokens). Therefore we do not create a validation set since we are in the infinite data regime and the training loss is an unbiased estimator of the validation loss (Hoffmann et al., 2022).

The size of the dataset D is defined as the number of total training tokens seen by a model during the full training. The corpus is tokenized by RoBERTa's Byte Pair Encoding (BPE) and the maximum sequence length is fixed at 1024 tokens.

Model. The model is a bidirectional attention Transformer (Vaswani et al., 2023) with some notable improvements/updates such as Rotary Positional Embeddings (RoPE) (Su et al., 2024; Grattafiori et al., 2024) and Gated Linear Units (Shazeer, 2020). See Appendix A for the detailed description of our architecture. Let N be the number of non-embedding parameters of the model, and F_N the non-embedding parameters FLOPS/token. In our study, 'the model size' is characterized by F_N (DeepSeek-AI et al., 2024) which for transformer encoder models is given by

$$\begin{aligned} F_N &= 6 \cdot (12 \cdot d^2 n_{layer}) + 12 \cdot n_{layer}\, s\, d \\ &= 6 \cdot N + 12 \cdot n_{layer} s\, d \end{aligned} \tag{1}$$

where s is the maximum sequence length, d the model's dimension and n_{layer} number of transformer blocks. The first term of Eq. (1) corresponds to the 2 FLOPS per parameter required during the forward pass of each matrix multiplication, and 4 FLOPS per parameter during the backward pass, totaling 6 FLOPS per parameter (N). The second term captures the cost of the attention mechanism, which involves two matrix multiplications: the computation of attention scores via $QK^\top$, and the application of these scores to the value matrix V. These operations are proportional to the sequence length and model's dimension. A detailed derivation of Eq. (1) is provided in Appendix B.

Compute C. Throughout our analysis/experiments we will use the relation:

$$C = F_N D \tag{2}$$

which was introduced in (DeepSeek-AI et al., 2024). Unlike the $C = 6ND$ estimation of (Kaplan et al., 2020; Hoffmann et al., 2022), Eq. (2) is more accurate because the second term in Eq. (1) takes into account the computational overhead of the attention operation. At low scales, this overhead is significant and cannot be neglected.

In section 4 we study the trade-off between dataset size D to model size F_N given by

$$R_{F_N}(C) = D/F_N. \quad (3)$$

which we label data-to-model ratio. The ratio offers insights on the scaling ability of BERT-like models and we compare such findings with current SOTA encoder models (Breton et al., 2025; Warner et al., 2024) and known scaling results of decoder models (Hoffmann et al., 2022; Grattafiori et al., 2024).

Objective. We adopt the standard BERT-style pretraining scheme based on the Masked Language Modeling (MLM) objective. Following NeoBERT (Breton et al., 2025), 20% of tokens in each input sequence are replaced with the special `[MASK]` token. Unless otherwise specified, all scaling experiments are conducted with a fixed masking rate. In Section 4.3, we investigate how the optimal masking percentage M varies with increasing compute.

Evaluation. We use training loss as the primary performance indicator for scaling. However, to assess transfer to downstream tasks, we also evaluate on benchmark suites. As GLUE (Wang et al., 2018) is largely saturated, we focus on MTEB (Muennighoff et al., 2022; Enevoldsen et al., 2025), while still reporting GLUE results for the models introduced in Section 5.

MTEB is a benchmark which measures the quality of text embeddings, requiring models to output a single representation per sequence of tokens. NeoBERT shows that MLM-pretrained encoders with naive[1] zero-shot poolings perform poorly without contrastive finetuning. Instead of adopting their full finetuning pipeline, we adopt a simpler alternative: an attentive pooling head trained using a supervised contrastive loss (InfoNCE (Conneau and Kiela, 2018)) on MNLI and SNLI training datasets. Despite its simplicity, this method achieves competitive results, underscoring the effectiveness of our pretraining. Appendix D provides further implementation details.

Methodology. All scaling experiments are based on the Meta Lingua codebase (Videau et al., 2024) and follow a common methodology. Given a set of interdependent variables (e.g., batch size and learning rate, or F_N and D) and a set of constraints (e.g., fixed compute C), we sample hundreds of valid parameter configurations. Each configuration is trained, and the final training loss, smoothed by averaging over 10 steps, is recorded. We then fit either a power law, a parabolic curve, or a parametric function to the runs and record the respective coefficients in Table 1.

[1] `[CLS]` or `mean-pooling`

	Var.	a	α
Power law $a \cdot C^{\alpha}$	ℓ_r	$1e(1.84 \pm 1.2)$	-0.24 ± 0.07
	bs	$1e(1.24 \pm 1.2)$	0.24 ± 0.07
	$^{\dagger}F_N$	$1e(0.22 \pm 0.39)$	0.46 ± 0.02
	$^{\dagger}D$	$1e(-0.22 \pm 0.39)$	0.54 ± 0.02
	$^{\dagger}R_{F_N}$	$1e(-0.44 \pm 0.78)$	0.08 ± 0.04
	$^{\star}F_N$	$1e(1.80 \pm 2.14)$	0.42 ± 0.09
	$^{\star}D$	$1e(-1.80 \pm 2.14)$	0.58 ± 0.09
	$^{\star}R_{F_N}$	$1e(-3.59 \pm 4.28)$	0.16 ± 0.18
	$^{\circ}L - E$	$1e(2.5 \pm 0.3)$	-0.125 ± 0.025
Parametric	$\hat{L}(F_N, D)$	$E = 1e(-0.36 \pm 0.3)$ $A = 1e(2.55 \pm 0.3)$, $\alpha = 0.326 \pm 0.04$ $B = 1e(2.26 \pm 0.4)$, $\beta = 0.236 \pm 0.05$	

Table 1: **Summary of fitted scaling laws** for different variables: lr learning rate, bs batch size, F_N non-embed parameters FLOPS/token, D total tokens, R_{F_N} data-to-model ratio, L training loss and $\hat{L}(F_N, D)$ Eq. (4). $\dagger$ denotes results derived in Section 4.1, $\star$ in Section 4.2 and $\circ$ from Fig.7.

3 Optimal learning rate and batch size

Method. We use Sobol sampling to select model size, total training tokens, learning rate, and batch size, as it ensures more uniform coverage of the four-dimensional space than random sampling, which can lead to uneven clustering. The sampled ranges are: 82M to 2.1B F_N, 250M to 10B total tokens D, learning rates between 5×10^{-4} and 10^{-2}, and batch sizes from 2^{16} to 2^{20}. For each sampled F_N value, we select the architecture whose non-embedding parameter FLOPS per token most closely matches the sampled value. Details of the matching procedure are provided in Appendix C.

Among the 512 sampled experiments, we select the top $k = 1$ points whose loss is minimal within a sliding window of size $[0.81C, C]$. We then fit a power law to these selected points to obtain the optimal learning rate and batch size with respect to compute C shown in Fig. 2.

Results. We note that the fits for learning rate and batch size have a large variance. This variability reflects the existence of multiple (lr, bs) pairs yielding similar performance for a fixed compute

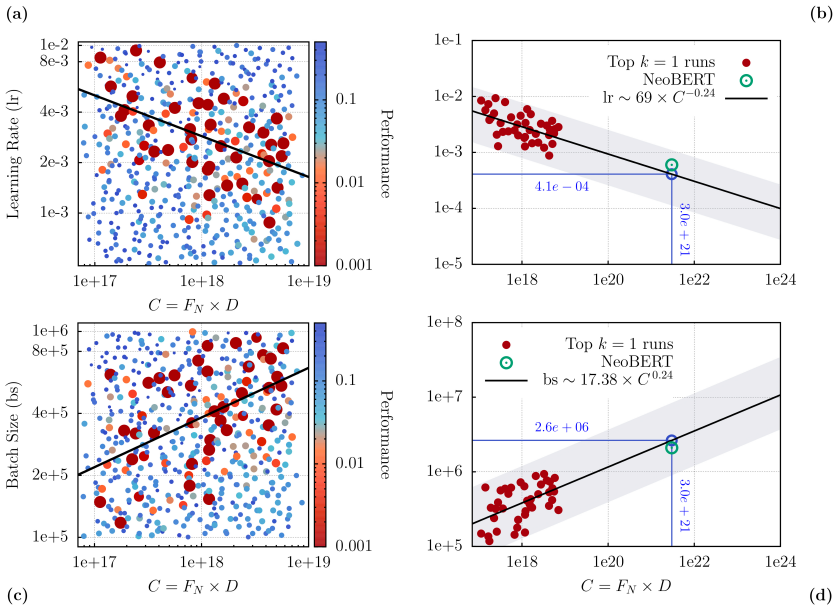


Figure 2: **Optimal Learning Rate and Batch size. (a, c)** Scatter plot of learning rate vs compute C for all training runs (resp. batch size vs compute C). We take a sliding window of size $[0.81C, C]$ and compute the lowest loss (bestloss) of the window. The loss of each point within the window is normalized by the bestloss. The log of this ratio determines Performance (the color/size of each point). Lower is better. The black line shows the fitted power-law relationship for top-1 performing runs. **(b, d)** Same plot as (a,c) but restricted to the top-k=1 run in each sliding window of C. The best learning rate follows the scaling law lr $\sim 69 \times C^{-0.24}$ (black line), whereas the batch size follows bs $\sim 17.38 \times C^{0.24}$. Blue annotations mark extrapolated predictions for optimal lr and bs for $C = 3.0e21$, which is the compute of NeoBERT (Breton et al., 2025).

(see Appendix E and Fig. 8), a phenomenon also reported in Li et al. 2024; DeepSeek-AI et al. 2024. There is no single best learning rate and batch size combination, but a range of admissible values in a relatively wide parameter space all yielding similar results. Hence any learning rate and batch size pair taken according to the scaling laws mentioned above will yield good results. Nonetheless, as compute increases, optimal training requires larger batch sizes and smaller learning rates, which is an expected behavior for the AdamW optimizer (Li et al., 2024).

4 Optimal data to model ratio

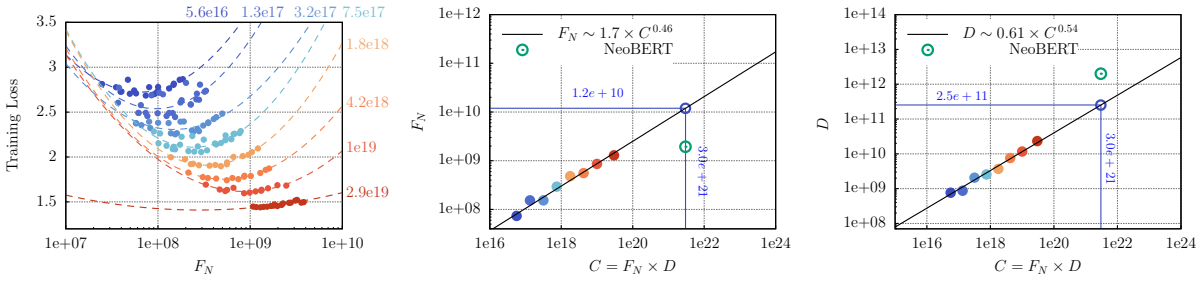
In this section we study how the data-to-model ratio, $R_{F_N} = D/F_N$ scales with compute. To this end we sample 8 logarithmically spaced FLOPS values ranging from $5e16$ to $2.9e19$. For each value we Sobol sample 16 to 21 data-to-model ratios R_{F_N} ranging from 1 to 100[2]. To extract the compute-optimal $R_{F_N}(C)$ we take two approaches, we fit a parabolic loss for every C (Section 4.1) or a parametric one (Section 4.2).

[2]except for the last iso-flop run where we restricted the range from 1 to 30

4.1 Parabolic fitting

Method. We fit a parabolic loss curve per FLOPS as shown on the leftmost panel of Fig. 3. For each such curves we take the run with the best training loss, i.e. the run closest to the minimum of the parabola, to obtain the optimal configuration. To which we then fit a power law $F_N = aC^{\alpha}$ and $D = bC^{\beta}$ shown in Fig. 3b and Fig. 3c respectively. We thus obtain the optimal ratio on how to scale model size and dataset with compute.

Results. For small F_N (i.e., smaller models), loss curves are unstable and noisy. As compute increases, the fits become smoother and the parabolas broaden, with reduced curvature. Indicating that the training becomes less sensitive to the data-to-model ratio. However, suboptimal ratios can still lead to divergence. Across compute budgets from 10^{17} to 10^{22}, we find the optimal ratio lies between 10 and 20. For comparison, decoder-only scaling laws typically use ratios $\lesssim 3$ (Kaplan et al., 2020; Besiroglu et al., 2024; DeepSeek-AI et al., 2024) with some exceptions such as Llama-3 (Dubey et al., 2024) and MiniCPM (Hu et al., 2024) which use a ratio of 7 and 32



(a) (b) (c)

Figure 3: **IsoFLOPS.(a)** For each compute value C we fit a parabola. Compute values are shown at the edges of the plot. **(b)** From (a), we select the lowest training loss run for a fixed C. We then fit a power law (black line) and extrapolate to get the optimal F_N Non-Embed Parameters FLOPS/tokens with respect to C. **(c)** Same as in (b) but we fit a power law and extrapolate to get the optimal Total Tokens D with respect to C. The blue lines illustrate an extrapolation point for $C = 3e21$ which is approx. the compute of NeoBERT (Breton et al., 2025). The final data to model ratio for a fixed C is $R_{F_N} = D/F_N \sim 0.36\, C^{0.08}$.

respectively. On the other hand, NeoBERT has $R_{F_N} \simeq 1400$, which is far from optimal.

4.2 Parametric loss fitting

Method. Although this approach derives equivalent results as the parabolic fitting, using the parametric loss we can extrapolate the training loss for any F_N and D and can therefore predict behaviors at larger computes. We use the same samples as in Section 4.1, but we fit the parametric loss introduced in Hoffmann et al. 2022 which we recall here:

$$\hat{L}(F_N, D) = E + \frac{A}{F_N^{\alpha}} + \frac{B}{D^{\beta}} \,. \tag{4}$$

However, instead of using L-BFGS algorithm as in Hoffmann et al. 2022 to minimize the Hubert loss:

$$\min_{\theta} \sum_{\text{runs } i} \text{Hub}_{\delta} \left(\log \hat{L}(F_N{}^{(i)}, D^{(i)}) - \log L^{(i)} \right) \,, \tag{5}$$

we take inspiration from Besiroglu et al. 2024 and use the BFGS algorithm instead, with $\delta = 10^{-3}$ to be less sensitive to outliers. $L^{(i)}$ is the training loss of the run i and θ are all the coefficients A, B, E, α, β of the parametric loss.

Results. A summary of the fitted coefficients and their variance is given in Table 1, the uncertainty was estimated by bootstrapping the samples 4000 times. The resulting curves are shown in Fig. 4. Note that contrary to the parabolic approach from Fig. 3 the fit remains good even for the last two profiles. The right panel of Fig. 4 shows an extrapolation of the parametric loss.

We now want to calculate the optimal data-to-model ratio from the fitted coefficients. From Eq. (4) enforcing a constant compute $C = F_N D$, we write the parametric loss as only a function of F_N i.e $\hat{L}(F_N, D) \equiv \hat{L}(F_N, C/F_N)$. Optimizing the loss with respect to F_N, yields the optimal F_N: $F_N(C) = n \cdot C^{\eta}$, where

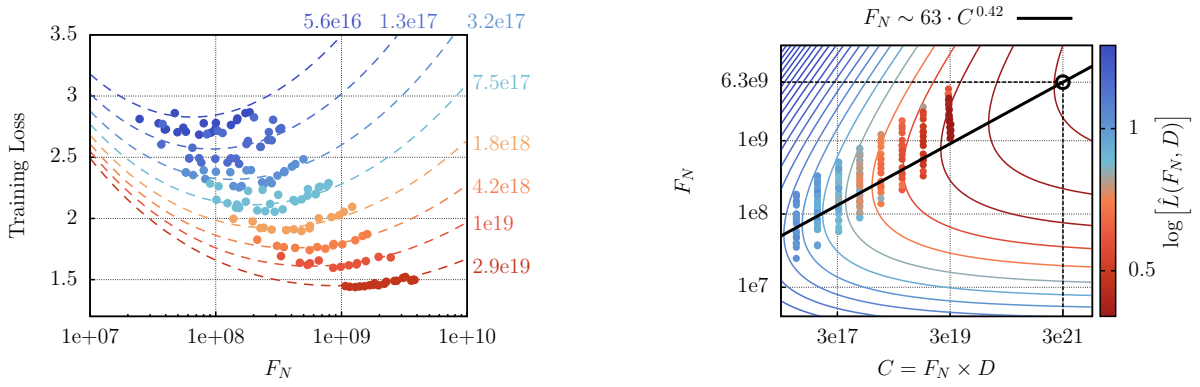
$$n = \left(\frac{A\alpha}{B\beta} \right)^{\frac{1}{\alpha+\beta}} \quad \text{and} \quad \eta = \frac{\beta}{\alpha+\beta} \,. \tag{6}$$

Identically, the compute-optimal dataset size is given by $D(C) = \frac{1}{n} \cdot C^{1-\eta}$. The data-to-model ratio is then given by

$$R_{F_N}(C) = \frac{D}{F_N} = \frac{1}{n^2} C^{1-2\eta} \,. \tag{7}$$

Fig. 5 plots the ratio with 80% confidence intervals, alongside the prediction from Chinchilla (Besiroglu et al., 2024). We can see that encoder models require 10-100× more data, or equivalently, 10-100× less parameters, than decoder models at fixed compute, consistent with prevailing assumptions in current research.

Finally, we verify experimentally whether small scale runs predict the behavior of large scale runs. The small scale runs correspond to the best performing models in Fig. 3a, ensuring that each point reflects a compute-optimal configuration. Since we do not expect the training loss L to reach zero, but to saturate at the incompressible value E, we fit a power law to $L - E_{\text{min}}$ and $L - E_{\text{max}}$, where E_{min} and E_{max} denote the bounds of the 95% confidence interval for E. In Fig. 7, we confirm that the training loss of a 1B parameter model trained optimally on 350B tokens is predicted by such scaling law. Hopefully, this allows future studies to



(a) (b)

Figure 4: **Parametric Fit. (a)** The iso-gamma profiles obtained from the fitted the parametric loss function given in Eq. (4) are plotted (with dashed lines) against the experimental results from our scaling runs. Compared to Fig. 3, the fit remains good even for the last two profiles. **(b)** A contour plot of the fitted parametric loss defined in Eq. (4) overlayed with the numerical results from our scaling runs. The black line shows the compute optimal scaling. The black dashed lines show an example compute optimal extrapolated run with $C = 3e21$.

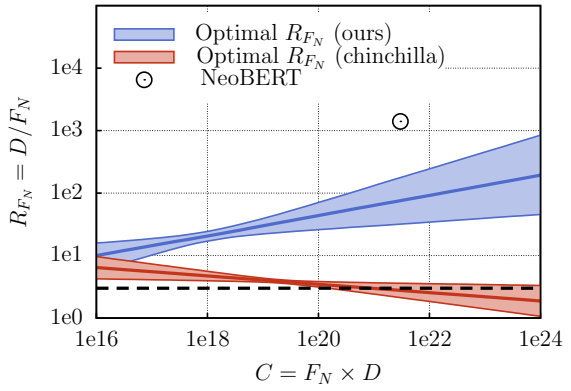


Figure 5: **Optimal data-to-model ratio** for encoder models (in blue) and for decoder models (in red). Shaded regions represent 80% confidence intervals. The black point corresponds to the training setup of NeoBERT, which is clearly far from optimal.

select compute-efficient pretraining configurations without resorting to costly grid searches.

4.3 Scaling laws of masking percentage

The Next Token Prediction (NTP) of decoders computes a loss on 100% of the input tokens. In contrast, MLM computes a loss only on the masked percentage M of tokens. Hence, the model receives gradient updates only from MD tokens during pretraining. Therefore, one may think that NTP scaling laws may hold for MLM if we consider the effective dataset size $D' = MD$ (Clark et al., 2020). This could indeed help explain the order of magnitude discrepancy between the optimal data-to-model ratios of decoder models ($\sim$ 3) compared to Section 4 ($\sim$ 10 - 100).

Method. To investigate this hypothesis we trained 128 models with Sobol sampled data-to-model ratios (from 1 to 100) and masking ratios (from 0.05 to 0.95) at 3 different compute budgets ($10^{17}, 3.2 \cdot 10^{17}$ and 10^{18}). Since the training loss directly depends on the masked percentage M-the hyperparameter under investigation- it cannot serve as a fair evaluation metric. Therefore, all models are finetuned and evaluated on `MTEB(eng, v2)` to enable consistent comparisons across runs.

Results. Fig. 6 shows that larger compute budgets seem to favor larger masking rates, with rates as high as 80% reaching astonishingly good downstream performance. Secondly, we see no evidence of a relationship between the masking ratio and the data-to-model ratio. Rejecting the hypothesis mentioned above, that the scaling laws of MLM simply behave as those of NTP on a re-scaled dataset.

5 OptiBERT

Current state-of-the-art BERT-style models are trained with very large data-to-model ratios R_{F_N}, NeoBERT uses approximately (1400), ModernBERT (810), DeBERTa-V3 (230) and NomicBERT (1100). Our analysis in Section 4 concludes that R_{F_N} should range from 10 to 100 for our compute budget. Hence current SOTA models are significantly over-trained, as can be seen in Fig. 5.

Therefore we re-train a NeoBERT equivalent

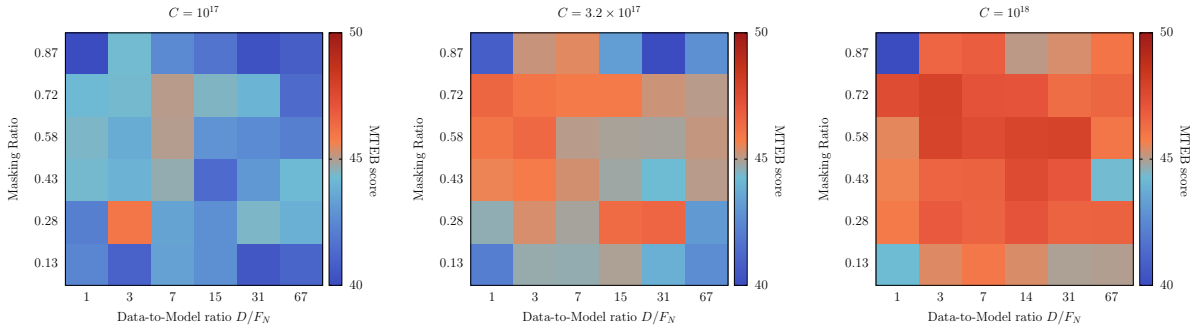


Figure 6: **Downstream performances on MTEB(eng, v2) for 3 different compute budgets.** The runs are binned into (6×6) grid.

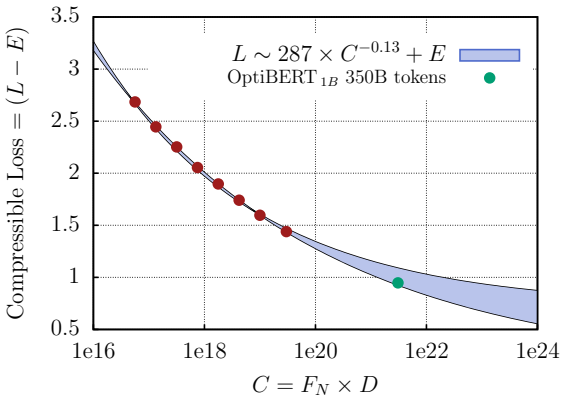


Figure 7: **Evolution of training loss with compute** The black line is the power law fitted on low-scale runs (red) with 95% confidence interval. The training loss of high-compute run (green) is perfectly predicted by scaling law. E is the coefficient in the parametric loss Eq. (4) given in Table 1.

architecture [3] on 1.3B, 13B, and 130B tokens reducing compute by factors of 2000, 200, and 20. We keep the same hyperparameters (batch size and learning rate) since we observe in Fig. 2 that they are optimal. We refer to these variants as OptiBERT$_{neo}$ and report results on `MTEB(eng, v1)` in Table 2 and on GLUE in Table 3. Results on `MTEB(eng, v2)` are provided in Table 5 in Appendix D.3. Our analysis focuses on Table 2 since the models are comparable, and the performance differences across MTEB versions is minimal.

Furthermore, we evaluate each of the 8 compute-optimal models in Fig. 3 (dubbed OptiBERT) on the `MTEB(eng, v1)` benchmark and show their average score in Fig. 1. In Table 2 and 3 we give only the detailed values of the 8th model i.e. the most compute-expensive one to compare with the performance of OptiBERT$_{neo}$.

[3] 28 layers, 768 hidden dimension and 12 attention head

5.1 Results

Our study of compute-optimality, in Section 4, showed that the training loss of OptiBERT models scales with compute. Figure 1 confirms that such scaling also applies to downstream MTEB performance. Indeed, our largest compute-optimal model, OptiBERT$_{239M}$, trained on 41B tokens, surpasses both NeoBERT and ModernBERT on the MTEB benchmark while requiring 50 times less compute. To put this in perspective, NeoBERT was trained for 6000 H100 GPU hours ($\sim$ 1 year) whereas our OptiBERT$_{239M}$ was trained in only 64 H100 GPU hours ($\sim$ 3 days).

Table 2 further supports the claim that current SOTA BERT-like models are over-trained. OptiBERT$_{neo}$ variants have the same model size and hyperparameter choices as NeoBERT but vary dataset size. The model trained on 130B tokens which is 20 times less than NeoBERT surpasses both NeoBERT and ModernBERT in MTEB score. Counter-intuitively, this shorter pre-training leads to stronger downstream performance at a fraction of the cost.

Note, however, that OptiBERT models underperform for Clustering, Retrieval and Reranking tasks compared to the SOTA models. The gap in performance could be as a result of context length: our models have a maximum sequence length of 1024, compared to 4096 for NeoBERT and 8192 for ModernBERT. Further extending the sequence length could close this gap but this is out of the scope of our current study.

Finally, Table 3 reports the GLUE performance of our models with their 95% confidence intervals computed by bootstraping 1000 GLUE

Model	Params	Tokens	Compute	Class.	Clust.	PairClass.	Rerank.	Retriev.	STS	Summ.	Avg.
$\text{ModernBERT}_{\text{large}}$	352M	2.019T	4.9e21	62.4	38.7	65.5	50.1	23.1	68.3	27.8	46.9
NeoBERT_{4096}	198M	2.1T	2.9e21	61.6	**40.8**	76.2	**51.2**	**31.6**	74.8	**30.7**	51.3
OptiBERT_{neo}	198M	1.3B	1.9e18	59.2	29.7	70.0	45.0	18.3	73.7	28.8	46.4
OptiBERT_{neo}	198M	13B	1.9e19	65.9	32.1	74.7	47.7	21.6	78.1	30.0	50.0
OptiBERT_{neo}	198M	130B	1.9e20	**67.5**	33.4	**77.0**	48.5	25.4	**81.1**	30.2	**51.9**
OptiBERT_{239M}	239M	41B	7.0e19	**67.5**	36.1	75.8	48.1	23.3	79.7	30.0	51.5

Table 2: **MTEB scores** on `MTEB(eng,v1)`. Baseline scores on ModernBERT and NeoBERT were retrieved from Table 4 of (Breton et al., 2025). Note these models have a context sequence length of 8192 and 4096 respectively whereas our runs are only 1024. Params refers to non-embedding parameters N (see Eq. (1)). **Tokens** denote the total number of training tokens the model has seen. **Compute** measure the amount of FLOPS required to pretrain the model.

Model	Params	Tokens	Compute	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	Avg.
$\text{ModernBERT}_{\text{large}}$	352M	2.019T	4.9e21	**90.8**	**95.2**	**92.7**	92.1	**97.1**	91.7	71.4	92.8	90.5
NeoBERT_{1024}	198M	2.0T	2.9e21	88.9	93.9	90.7	91.0	95.8	93.4	64.8	92.1	88.8
OptiBERT_{neo}	198M	1B	1.9e18	$77.9^{\uparrow 78.7}_{\downarrow 76.9}$	$84.7^{\uparrow 86.1}_{\downarrow 83.5}$	$87.4^{\uparrow 87.9}_{\downarrow 86.9}$	$68.4^{\uparrow 74.7}_{\downarrow 62.1}$	$89.2^{\uparrow 92.1}_{\downarrow 86.5}$	$85.8^{\uparrow 89.3}_{\downarrow 81.9}$	$28.5^{\uparrow 36.8}_{\downarrow 21.0}$	$86.2^{\uparrow 87.7}_{\downarrow 84.6}$	$76.0^{\uparrow 79.2}_{\downarrow 72.9}$
OptiBERT_{neo}	198M	13B	1.9e19	$85.2^{\uparrow 86.0}_{\downarrow 84.4}$	$91.0^{\uparrow 91.9}_{\downarrow 90.0}$	$89.8^{\uparrow 90.2}_{\downarrow 89.4}$	$81.4^{\uparrow 86.6}_{\downarrow 75.8}$	$92.1^{\uparrow 94.3}_{\downarrow 89.8}$	$90.6^{\uparrow 94.2}_{\downarrow 87.0}$	$56.6^{\uparrow 64.3}_{\downarrow 48.1}$	$90.1^{\uparrow 91.3}_{\downarrow 88.7}$	$84.6^{\uparrow 87.4}_{\downarrow 81.6}$
OptiBERT_{neo}	198M	130B	1.9e20	$89.0^{\uparrow 89.6}_{\downarrow 88.5}$	$93.7^{\uparrow 94.5}_{\downarrow 92.7}$	$90.1^{\uparrow 91.1}_{\downarrow 88.4}$	$88.7^{\uparrow \mathbf{93.1}}_{\downarrow 84.5}$	$94.6^{\uparrow 96.2}_{\downarrow 92.7}$	$92.4^{\uparrow \mathbf{95.5}}_{\downarrow 89.1}$	$64.6^{\uparrow \mathbf{71.9}}_{\downarrow 57.6}$	$92.0^{\uparrow \mathbf{92.9}}_{\downarrow 91.0}$	$88.1^{\uparrow \mathbf{90.6}}_{\downarrow 85.6}$
OptiBERT_{239M}	239M	41B	7.0e19	$86.6^{\uparrow 87.2}_{\downarrow 85.8}$	$92.1^{\uparrow 93.0}_{\downarrow 91.2}$	$90.3^{\uparrow 90.7}_{\downarrow 89.9}$	$83.2^{\uparrow 88.4}_{\downarrow 77.6}$	$92.6^{\uparrow 94.6}_{\downarrow 90.3}$	$91.0^{\uparrow 94.2}_{\downarrow 87.7}$	$59.6^{\uparrow 66.4}_{\downarrow 50.8}$	$90.8^{\uparrow 91.9}_{\downarrow 89.4}$	$85.8^{\uparrow 88.3}_{\downarrow 82.8}$

Table 3: **GLUE scores** on the validation set. Baseline scores were retrieved from Table 3 of NeoBERT (Breton et al., 2025), from Table 5 of ModernBERT (Warner et al., 2024). **Params** (N) refers to non-embedding model parameters. **Tokens** denote the total number of training tokens the model has seen. **Compute** measures the amount of FLOPS required to pretrain the model. The mean scores of our models are shown with their 95% confidence intervals obtained by bootstraping 1000 times the GLUE evaluations over 5 different seeds.

evaluations over 5 different seeds. Based on such intervals we can observe that the GLUE benchmark is saturated with fluctuations so large that we cannot significantly differentiate the performance of the top 3 models in the table. OptiBERT_{neo}'s performance is in the range $(85.6, 90.6)$ which includes both ModernBERT's 90.5 and NeoBERT's 88.8 scores. Hence why our analysis is mainly based on MTEB performance instead of GLUE.

6 Related Work

BERT-based Models. Encoder models are a leaner and cheaper alternative to decoder models; requiring significantly lower training costs whilst remaining powerful tools for language modeling tasks such as semantic similarity, information retrieval, clustering, and classification (Cer et al., 2018; Muennighoff et al., 2022). Since the introduction of BERT, a wide range of encoder models (Devlin et al., 2019; Liu et al., 2019a; He et al., 2020, 2021; Conneau et al., 2019) have been developed and serve as backbones for various NLP tasks. Since then, research on encoder models has focused on either finetuning existing backbones by large-scale contrastive learning or improving the pretraining and architecture of such backbones. GTE (Li et al., 2023), CDE (Morris and Rush, 2024), Jina Embeddings (Sturua et al., 2024), DPR (Karpukhin et al., 2020), Contriever (Izacard et al., 2021), Sentence-BERT (Reimers and Gurevych, 2019) and LaBSE (Feng et al., 2020) are some of the models focused on finetuning, whilst ModernBERT (Warner et al., 2024) and NeoBERT (Breton et al., 2025) focus on the pretraining by increasing dataset size to trillions of tokens and architectural improvements. However in our study we have shown that blindly increasing data can be extremely inefficient and does not result in significant gains. Much better performance can be obtained at a fraction of the cost by increasing data and model size in tandem, while keeping a data-to-model ratio R_{F_N} around 10 to 100.

Scaling Laws. Neural scaling laws describe how model performance improves with increased compute, data, and parameter size. First introduced by Hestness et al. 2017 and subsequently developed for decoder models (Kaplan et al., 2020; Hoffmann et al., 2022; DeepSeek-AI et al., 2024), they have since been extended to other architectures,

including Mixture-of-Experts (Ludziejewski et al., 2024, 2025; Abnar et al., 2025; Liew et al., 2025) and, more recently, encoders (Ivgi et al., 2022; Urbizu et al., 2023a; Fang et al., 2024). Urbizu et al. (2023a) explore scaling laws for BERT in the low-resource regime and observe very different trends from decoder-based laws. However, their study is limited to three model sizes. Our work established broader and more predictive scaling laws for large scale pretraining. Consistently with the results from Urbizu et al. 2023a we find that decoder-based laws do not generalize to encoders. We have seen that encoders are 10 to 100 times more data-hungry than decoders and resp. 10 to 100 times more parameter-efficient than decoders.

Hyperparameter Estimation. Beyond model and dataset size, pretraining performance also depends on hyperparameters such as learning rate, batch size, optimizer settings (Ali et al., 2023; Hägele et al., 2024; DeepSeek-AI et al., 2024; Porian et al., 2024) and the masking percentage of the MLM objective (Wettig et al., 2022; Ankner et al., 2023). Inspired by Porian et al. 2024 and Hägele et al. 2024, we have kept the scheduler fixed to a cosine decay, systematically exploring only the learning rate and batch size to understand their effect under a fixed compute budget as in DeepSeek-AI et al. (2024). Secondly, Wettig et al. 2022 showed that BERT's original 15% masking[4] is not always a good choice. They suggest that larger models benefit from a higher masking. Following their results we have trained the majority of our experiments with 20% masking. In section 4.3 of the paper we saw that larger compute budgets favor larger masking ratios with optimal masking ratios ranging from 20% to 50%.

7 Ethical Considerations

We acknowledge that our experiments required significant computational resources which we estimate a total of 4700 H100 GPU-hours. Code optimizations yielded 300 TFLOP/s per GPU on average. Our largest run with the same compute as NeoBERT completed in 2600 H100 GPU-hours, compared with NeoBERT's 6000. By identifying compute-optimal configurations, we hope that our findings can lower the environmental and financial costs of future pretraining.

8 Limitations

Our study adopts a fixed dataset for all scaling experiments. While `FineWeb-edu` is a high-quality English corpus, prior work (DeepSeek-AI et al., 2024) has demonstrated that both the choice and quality of the dataset can significantly influence the conclusions drawn from scaling law analyses. As such, the generality of our findings may not extend to low-quality corpora or low-resource language regimes. Investigating how the data-to-model ratio evolves with increasing compute under such conditions has been studied in Urbizu et al. 2023b but remains an important direction for future research.

Furthermore, our study focuses on a limited set of hyperparameters: learning rate, batch size, and masking ratio, while holding others fixed. Additional factors such as the choice of optimizer, learning rate scheduler, tokenizer, and architectural design can significantly influence performance. For instance, we fix the width-to-depth ratio[5] to 64, following BERT-base, though prior work suggests that model performance can be sensitive to this ratio (Levine et al., 2020).

We focus on scaling behavior with respect to training loss, but provide limited downstream evaluation. A more comprehensive study of downstream task performance across compute budgets could help identify emergent capabilities in transformer encoders trained with the MLM objective (Wei et al., 2022; Zhang et al., 2020).

Finally, both GLUE and MTEB benchmarks require a finetuning stage before evaluation. In GLUE, this involves task-specific finetuning, while in MTEB, a contrastive learning finetuning to pool the embeddings per token into sentence embeddings. Without such a step the models would get a bad performance because the MLM objective does not correspond to that of downstream tasks. However, this additional finetuning step makes it difficult to isolate the impact of pretraining alone.

Moreover, for larger size models (>1B) our simple finetuning with SNLI and MNLI is too restrictive as the datasets would be too small compared to the parameters of the model. This would result in overfitting, which would not give a fair evaluation of the pretraining.

[4] 80% of tokens are replaced with `[MASK]`, 10% with a random token and the remaining 10% are unchanged.

[5] the width-to-depth ratio is the model's hidden dimension over the number of layers

References

Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. 2025. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*.

Yasser A Ali, Emad Mahrous Awwad, Muna Al-Razgan, and Ali Maarouf. 2023. Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*, 11(2):349.

Zachary Ankner, Naomi Saphra, Davis Blalock, Jonathan Frankle, and Matthew L Leavitt. 2023. Dynamic masking rate schedules for mlm pretraining. *arXiv preprint arXiv:2305.15096*.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, and 1 others. 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.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. 2024. Chinchilla scaling: A replication attempt. *arXiv preprint arXiv:2404.10102*.

Lola Le Breton, Quentin Fournier, Mariam El Mezouar, and Sarath Chandar. 2025. Neobert: A next-generation bert. *arXiv preprint arXiv:2502.19587*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, and 1 others. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, and 69 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *Preprint*, arXiv:2401.02954.

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, volume 1 (long and short papers)*, pages 4171–4186.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. 2024. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. Mmteb: Massive multilingual text embedding benchmark. *Preprint*, arXiv:2502.13595.

Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. 2024. Scaling laws for dense retrieval. *Preprint*, arXiv:2403.18684.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, and 1 others. 2024. Scaling

laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep learning scaling is predictable, empirically. *Preprint*, arXiv:1712.00409.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, and 1 others. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Maor Ivgi, Yair Carmon, and Jonathan Berant. 2022. Scaling laws under the microscope: Predicting transformer performance from small scale experiments. *Preprint*, arXiv:2202.06387.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and 1 others. 2024. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

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*, pages 39–48.

Yoav Levine, Noam Wies, Or Sharir, Hofit Bata, and Amnon Shashua. 2020. Limits to depth efficiencies of self-attention. *Advances in Neural Information Processing Systems*, 33:22640–22651.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, and 1 others. 2024. Surge phenomenon in optimal learning rate and batch size scaling. *arXiv preprint arXiv:2405.14578*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Seng Pei Liew, Takuya Kato, and Sho Takase. 2025. Scaling laws for upcycling mixture-of-experts language models. *arXiv preprint arXiv:2502.03009*.

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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król,

Tomasz Odrzygóźdź, Piotr Sankowski, and 1 others. 2024. Scaling laws for fine-grained mixture of experts. In *Forty-first International Conference on Machine Learning*.

Jan Ludziejewski, Maciej Pióro, Jakub Krajewski, Maciej Stefaniak, Michał Krutul, Jan Małaśnicki, Marek Cygan, Piotr Sankowski, Kamil Adamczewski, Piotr Miłoś, and 1 others. 2025. Joint moe scaling laws: Mixture of experts can be memory efficient. *arXiv preprint arXiv:2502.05172*.

John X Morris and Alexander M Rush. 2024. Contextual document embeddings. *arXiv preprint arXiv:2410.02525*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. 2024. Resolving discrepancies in compute-optimal scaling of language models. *Advances in Neural Information Processing Systems*, 37:100535–100570.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

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

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and 1 others. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173*.

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

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*.

Gorka Urbizu, Iñaki San Vicente, Xabier Saralegi, Rodrigo Agerri, and Aitor Soroa. 2023a. Scaling laws for BERT in low-resource settings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7771–7789, Toronto, Canada. Association for Computational Linguistics.

Gorka Urbizu, Iñaki San Vicente, Xabier Saralegi, Rodrigo Agerri, and Aitor Soroa. 2023b. Scaling laws for BERT in low-resource settings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7771–7789, Toronto, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. 2024. Meta Lingua: A minimal PyTorch LLM training library.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, and 1 others. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2022. Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Yian Zhang, Alex Warstadt, Haau-Sing Li, and Samuel R Bowman. 2020. When do you need billions of words of pretraining data? *arXiv preprint arXiv:2011.04946*.

A Architecture/Implementation Details

In this section, we explain the architectural details, data processing, optimization strategy, and pretraining procedure that are shared across all experiments. Unless otherwise specified, these settings remain fixed and serve as the reference configuration for all reported results.

Transformer Block. Our architecture follows recent design choices introduced in NeoBERT (Breton et al., 2025) and ModernBERT (Warner et al., 2024). Each transformer block consists of a self-attention block followed by a feed-forward network employing a Gated Linear Unit with Swish activation (SwiGLU) (Shazeer, 2020). The hidden dimension of the SwiGLU is set to 2/3 of the model dimension to match the number of learnable parameters of a standard feed-forward block. Finally, all bias terms are removed from the model.

Normalization. We use a pre-norm Transformer architecture (Xiong et al., 2020) with Root Mean Square Layer Normalization (RMSNorm) (Zhang and Sennrich, 2019). Since all biases are removed from the network, RMSNorm is sufficient to stabilize training while being more computationally efficient and parameter-light than standard layer normalization (Ba et al., 2016). To further improve stability, we apply an additional RMSNorm layer directly after the token embedding layer, before the Transformer blocks (Takase et al., 2023).

Data. All models are trained on the `Fineweb_edu` dataset (Penedo et al., 2024). We apply sequence packing without padding and restrict attention to tokens within the same sequence. Rotary Positional Embeddings (RoPE) are used for position encoding (Su et al., 2024), and tokenization is performed with the RoBERTa Byte Pair Encoding (BPE) tokenizer (Liu et al., 2019b), using a vocabulary of 50,265 tokens. The maximum sequence length is fixed at 1024 tokens. The total batch size varies according to the scaling law regime detailed in Section 3.

Optimization. We use the AdamW optimizer (Loshchilov and Hutter, 2017) with a fixed weight decay of 0.1, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping with a threshold of 1. The learning rate is linearly warmed up over a fixed number of steps and subsequently decayed to 10% of its peak value following a cosine schedule. To maintain consistent warmup across different training regimes, the warmup phase covers 10% of the total training steps, constrained between 500 and 50,000 steps.

Efficiency. We adopt FlexAttention (Dong et al., 2024), sequence packing and `torch.compile` to maximize computational efficiency (Ansel et al., 2024). Code is based on Meta Lingua (Videau et al., 2024).

B Details on the non-embed parameters FLOPS/tokens F_N

We recall Eq. 1, which expresses the number of FLOPs per token attributed to non-embedding parameters:

$$\begin{aligned} F_N &= 6 \cdot (12 \cdot d^2 \, n_{\text{layer}}) + 12 \cdot n_{\text{layer}} s d \\ &= 6N + 12 \cdot n_{\text{layer}} s \, d \, , \end{aligned}$$

where d is the model dimension, n_{layer} the number of transformer layers, and s the maximum sequence length.

The cost of a matrix multiplication is 2FLOPS (addition, multiplication) during a forward pass and 4FLOPS during a backward pass. The first term accounts for the cost of forward and backward passes through the model's non-embedding parameters N. Each transformer layer contains four $(d \times d)$ projections for attention (Q, K, V, O) and three $(d \times d_{\text{FFN}})$ projections in the feed-forward network, with $d_{\text{FFN}} = \frac{2}{3} \cdot 4d$. This results in $12d^2$ FLOPS per layer per pass, and 6 total passes (forward and backward over both attention and feed-forward), yielding $6 \cdot 12d^2 n_{\text{layer}} = 6N$.

The second term represents the attention overhead, as discussed in DeepSeek-AI et al. (2024). Each token attends to s others, leading to s^2 attention scores per layer. Each dot product between d-dimensional queries and keys requires $2d$ FLOPs, with an additional $2d$ for applying the values. The total is thus $4d$ FLOPs per pair in the forward pass and $8d$ in the backward pass, or $12d$ FLOPs in total. Repeating this across n_{layer} layers and normalizing by s tokens gives the second term, $12 n_{\text{layer}} s d$.

C Details on the matching algorithm

When performing scaling laws, we need to create a wide variety of encoder architectures with different randomly sampled F_N counts where F_N is the non-embed parameter FLOPS/token count. For

a given F_N there can be multiple architectures, because we can 'play' with a few key variables:

- The hidden dimension d, i.e. the width of the model
- The number of layers n_{layer}, i.e. the depth of the model
- The number of key-value heads $n_{\text{kv heads}}$
- The number of query heads $n_{\text{q heads}}$
- The inner dimension d_{ffn} of the GLU blocks

The choice whether an architecture is more deep or more wide depends on such variables. However there are certain combinations (1 layer deep model with a huge dimension) that do not make sense. Therefore we impose the following hard constraints to the sampling of the above variables:

1. The hidden dimension d has to be a multiple of the number of query heads $n_{\text{q heads}}$. Hence, instead of sampling d and $n_{\text{q heads}}$ we will sample $n_{\text{q heads}}$ and an integer dimension-per-head ratio $r_{\text{dim/head}}$ and subsequently define $d = r_{\text{dim/head}} \cdot n_{\text{q heads}}$. This constraint is imposed by the functioning of the multi-head-attention block.
2. The inner dimension d_{ffn} of the GEGLU blocks is kept fixed at

$$d_{\text{ffn}} = 256 \cdot \left\lceil \frac{2/3 \cdot 4d}{256} \right\rceil . \tag{8}$$

   The ratio of $2/3$ is chosen to keep the parameter count of a GEGLU block equal an equivalent MLP feed-forward block. The constant expansion factor of 4 is chosen inline with common best practice for transformer architectures.

As well as a couple of soft constraints

3. The width-to-depth ratio $r_1 = d/n_{\text{layer}}$ is kept close to 64. This value was chosen to stay close to the original dim-per-layer ratio of BERT. We must set it to some typical value to keep runs comparable since this ratio has some influence on the performance of the encoder (Levine et al., 2020).
4. The ratio of query heads to key-value heads $r_2 = n_{\text{q heads}}/n_{\text{kv heads}}$ is kept as low as possible, i.e. close to 1. Similarly, this ratio controls the behavior of the attention mechanism and hence must also be regulated to keep different runs comparable.
5. The non-embed parameter FLOPS per token count (calculated via Eq.1) of the sampled architecture must match as closely as possible to the target F_N count.

Each soft constraint is applied by minimizing the following score over all possible variable combinations:

$$\begin{aligned} \text{score} =& 5|\log(F_N) - \log(F_{N\text{target}})| \\ &+ 0.05\frac{|r_1 - 64|}{64} \\ &+ 0.05|r_2 - 1| , \end{aligned} \tag{9}$$

where $F_{N\text{target}}$ is the desired non-embed parameter FLOPS/token count, $r_1 = d/n_{\text{q head}}$ and $r_2 = n_{\text{q heads}}/n_{\text{kv heads}}$.

D MTEB and GLUE benchmark

D.1 GLUE hyperparameters

We follow the exact finetuning procedure described in the NeoBERT's Appendix C Breton et al. 2025 We search over learning rates {5e-6, 6e-6,1e-5,2e-5,3e-5} and batch sizes {4,8,16,32} and weight-decay {1e-2, 1e-5} and epochs {2, 10}. In Table 4 are the hyperparameters that yielded the best GLUE score per task.

D.2 Fine-tuning task before MTEB

The recent Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2022) and its subsequent second iteration (MTEBv2) (Enevoldsen et al., 2025) are more complete and challenging benchmarks to measure the quality of the embeddings.

However, MTEB evaluates sentence embedding models, i.e. models which output a single embedding for a chunk of text, unlike transformer encoder models trained with MLM which produce an embedding per token. Hence, to evaluate on MTEB we need to teach the encoder model how to pool the embeddings of all tokens into a single meaningful embedding. A multitude of techniques (Li et al., 2023; Sturua et al., 2024; Morris and Rush, 2024) have been developed to turn encoders into strong sentence embedders, but in order to keep the focus on the quality of the

Model	Task	bs	lr	wd	ep
OptiBERT$_{neo}$	MNLI	16	3e−5	1e−5	2
	QNLI	16	2e−5	1e−2	10
1.3B tokens	QQP	16	3e−5	1e−5	2
	RTE	8	6e−6	1e−2	10
	SST	4	6e−6	1e−2	10
	MRPC	8	3e−5	1e−5	10
	CoLA	4	2e−5	1e−5	10
	STSB	8	3e−5	1e−5	10
OptiBERT$_{neo}$	MNLI	32	3e−5	1e−5	2
	QNLI	4	1e−5	1e−2	10
13B tokens	QQP	16	2e−5	1e−2	2
	RTE	8	2e−5	1e−2	10
	SST	16	3e−5	1e−5	10
	MRPC	32	2e−5	1e−2	10
	CoLA	4	1e−5	1e−2	10
	STSB	16	3e−5	1e−2	10
OptiBERT$_{neo}$	MNLI	32	1e−5	1e−2	2
	QNLI	16	6e−6	1e−5	10
130B tokens	QQP	8	6e−5	1e−2	2
	RTE	8	5e−6	1e−2	10
	SST	16	5e−6	1e−5	10
	MRPC	16	1e−5	1e−5	10
	CoLA	4	5e−6	1e−5	10
	STSB	4	5e−6	1e−2	10

Table 4: Batch size (bs), learning rate (lr), weight decay (wd) and epochs (ep) used for fine-tuning OptiBERT models of different dataset sizes on GLUE tasks.

MLM pretraining we adopt the simplest finetuning which was introduced by SimCSE (Gao et al., 2021).

We add an attentive pooling head at the end of the encoder and we finetune the model end-to-end on MNLI and SNLI datasets following the supervised SimCSE prescription. The MNLI and SNLI datasets consists of triplets (x_i, x_i^+, x_i^-) where x_i and x_i^+ are 'positive' pairs, i.e. sequences containing similar information, whereas (x_i, x_i^-) are 'negative' pairs, i.e. sequences which may appear similar but contain incompatible information. Each sequence is passed through the model to obtain embeddings (h_i, h_i^+, h_i^-) and the model is finetuned using a contrastive InfoNCE loss

$$\mathcal{L} = -\log \frac{e^{\mathrm{sim}(h_i, h_i^+)/\tau}}{\sum_{j=1}^{B} \left(e^{\mathrm{sim}(\mathrm{h_j}, \mathrm{h_j^+})/\tau} + e^{\mathrm{sim}(h_j, h_j^-)/\tau} \right)} \tag{10}$$

where B is the batch size, i.e. the number of triplets in a forward pass. As in SimCSE we finetune for 3 epochs over MNLI and SNLI.

D.3 MTEB(eng,v2)

Table 2 reports results on `MTEB(eng, v1)` to enable comparison with NeoBERT and ModernBERT, which use this version. For completeness and to align with the recent convention of reporting `MTEB(eng, v2)` scores, we also provide results in Table 5 for future reference.

E Learning rate vs Batch size plots

Fig. 8 plots the relation between learning rate and batch size for a fixed compute budget for different runs. We observe that there are multiple runs with different (lr, bs) pairs that have good performance. There is more of an 'area' optimality rather than a specific point. This wide range of optimal hyperparameters combinations, instead of a single unique optimum, explains the noisiness of the fit in Fig. 2.

Model	Params	Tokens	Compute	Class.	Clust.	PairClass.	Rerank.	Retriev.	STS	Summ.	Avg.
OptiBERT$_{neo}$	198M	1.3B	1.9e18	64.6	37.9	70.0	39.3	21.7	73.3	24.7	47.4
OptiBERT$_{neo}$	198M	13B	1.9e19	70.1	39.1	74.7	41.0	24.4	77.8	26.2	50.5
OptiBERT$_{neo}$	198M	130B	1.9e20	71.5	39.0	**77.0**	**41.1**	**29.1**	**81.0**	**29.4**	**52.6**
OptiBERT$_{17M}$	17M	870M	1.3e17	62.9	37.8	65.9	38.9	18.0	70.0	25.4	45.6
OptiBERT$_{17M}$	17M	2B	3.2e17	63.5	38.3	67.4	39.3	20.1	71.7	25.1	46.5
OptiBERT$_{38M}$	38M	2.5B	7.5e17	66.1	37.1	70.7	39.1	16.2	72.9	26.3	46.9
OptiBERT$_{28M}$	28M	4.7B	1.2e18	65.6	37.6	69.4	39.9	17.9	73.7	28.5	47.5
OptiBERT$_{76M}$	76M	7.6B	4.2e18	69.1	**39.3**	72.9	40.5	21.5	75.6	28.6	49.6
OptiBERT$_{120M}$	120M	10B	1e19	70.1	38.6	73.6	40.4	26.3	77.8	28.6	50.8
OptiBERT$_{182M}$	182M	23B	3.0e19	71.3	38.9	76.7	40.9	21.5	78.7	27.5	50.8
OptiBERT$_{239M}$	239M	41B	7.0e19	**71.9**	37.9	75.8	40.7	26.3	79.4	28.7	51.6

Table 5: **MTEB(eng, v2) scores**. **Params** refers to non-embedding parameters N (see Eq. (1)). **Tokens** denote the total number of training tokens the model has seen. **Compute** measure the amount of FLOPS required to pretrain the model.

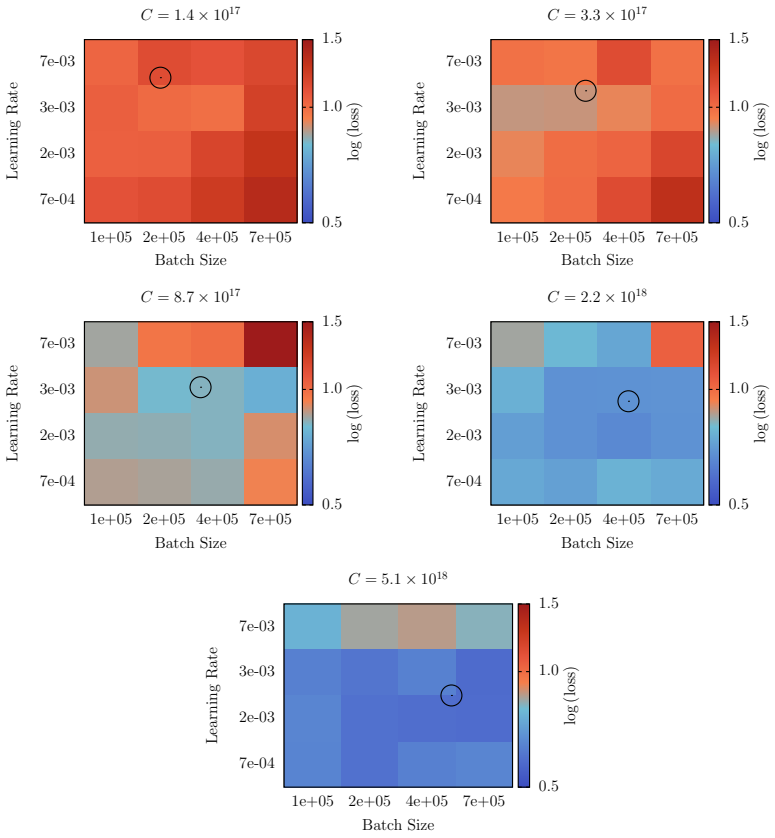


Figure 8: **Learning rate vs Batch size for a fixed compute**. From left to right we increase compute C (value shown above each plot). In black circle we depict the predicted compute optimal according to the scaling law in Fig.2. The runs are binned into a (4×4) grid and the colors show the log(loss).