What Language Model to Train if You Have One Million GPU Hours?

The BigScience Architecture & Scaling Group

Teven Le Scao[1*] **Thomas Wang**[1*] **Daniel Hesslow**[2*] **Lucile Saulnier**[1*] **Stas Bekman**[1*]
M Saiful Bari[3] **Stella Biderman**[4,5] **Hady Elsahar**[6] **Jason Phang**[7] **Ofir Press**[8] **Colin Raffel**[1]
Victor Sanh[1] **Sheng Shen**[9] **Lintang Sutawika**[10] **Jaesung Tae**[1] **Zheng Xin Yong**[11]
Julien Launay[2,12†] **Iz Beltagy**[13†]

[1] Hugging Face [2] LightOn [3] NTU, Singapore [4] Booz Allen [5] EleutherAI [6] Naver Labs Europe [7] New York University
[8] University of Washington [9] Berkeley University [10] Big Science [11] Brown University [12] LPENS [13] Allen Institute for AI

Abstract

The crystallization of modeling methods around the Transformer architecture has been a boon for practitioners. Simple, well-motivated architectural variations may transfer across tasks and scale, increasing the impact of modeling research. However, with the emergence of state-of-the-art 100B+ parameters models, large language models are increasingly expensive to accurately design and train. Notably, it can be difficult to evaluate how modeling decisions may impact emergent capabilities, given that these capabilities arise mainly from sheer scale. Targeting a multilingual language model in the 100B+ parameters scale, our goal is to identify an architecture and training setup that makes the best use of our 1,000,000 A100-GPU-hours budget. Specifically, we perform an ablation study comparing different modeling practices and their impact on zero-shot generalization. We perform all our experiments on 1.3B models, providing a compromise between compute costs and the likelihood that our conclusions will hold for the target 100B+ model. In addition, we study the impact of various popular pretraining corpora on zero-shot generalization. We also study the performance of a multilingual model and how it compares to the English-only one. Finally, we consider the scaling behaviour of Transformers to chose the target model size, shape, and training setup.

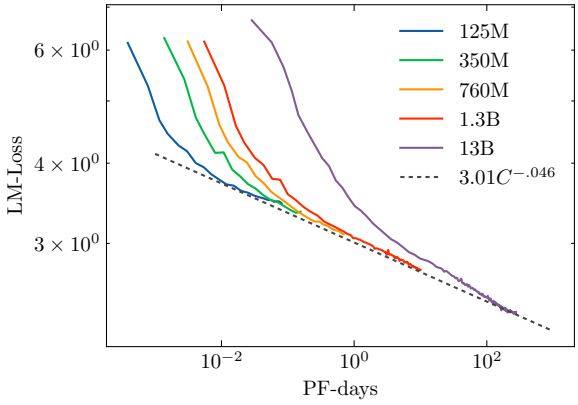

Figure 1: **Smooth scaling of language modeling loss as compute budget and model size increase**. We observe a power-law coefficient $\alpha_C \sim 0.046$, in-line with previous work. We use this to estimate the optimal model size and number of tokens for the available budget.

1 Introduction

Recent years have seen the advent of so-called foundation models (Bommasani et al., 2021), with emergent capabilities (e.g., zero-shot generalization) arising from sheer scale alone (Radford et al., 2019; Brown et al., 2020). Increasingly large language models are better models, predictably so: simple scaling laws connect the number of parameters, pretraining dataset size, and compute budget (Kaplan et al., 2020), providing a clear path towards more capable models. This paradigm shift can be attributed to the wide adoption of the Transformer (Vaswani et al., 2017), providing a scalable basis for practitioners to build upon.

In this paper, we design an architecture and training setup for a multilingual 100B+ parameters model, seeking to best use a fixed 1,000,000 A100-hours budget. Many promising additions and tweaks have been proposed to the Transformer architecture (Press et al., 2022; Shazeer, 2020; Dettmers et al., 2021) but it is difficult to identify which will best transfer to increased scales, and which are idiosyncrasies of specific setups.

Contributions. We study the impact of pretraining corpora, positional embeddings, activation functions, and embedding norm on zero-shot generalization. We base our study on the popular GPT-3 architecture (Brown et al., 2020), with experiments at the 1.3B parameters scale. We consider the impact of massive multilinguality, and, finally, we study the scaling of our models, and draft an architecture for a 176B model from our findings.

[*] Equal contribution.
[†] Equal supervision.

2 Methods

We first justify our choice to base our model on the popular recipe of combining a decoder-only model with an autoregressive language modeling objective, and introduce our experimental setup. We then discuss our evaluation benchmarks, and motivate our choice of zero-shot generalization as our key metric. Finally, we introduce the baselines we compare to throughout the paper.

2.1 Architecture and Pretraining Objective

In this paper, we base all models on a decoder-only Transformer pretrained with an autoregressive language modeling objective. This is a popular choice for large language models (Brown et al., 2020; Rae et al., 2021; Thoppilan et al., 2022), possibly because it lends itself to zero-shot application to many downstream tasks (Radford et al., 2019). Alternatives include encoder-decoder models trained with a span-corruption objective (e.g., T5 Raffel et al. (2019)), as well as non-causal decoders models with visibility over a prefix (so-called PrefixLMs, Liu et al. (2018); Dong et al. (2019)).

Our decision is motivated by the findings of Anonymous (2022), which showed that decoder-only models combined with an autoregressive language modelling objective provide the best zero-shot generalization abilities right after pretraining. Although multitask finetuning (Sanh et al., 2021) will favor an encoder-decoder with span corruption for best zero-shot generalization, they also found a compromise between these two practices. Following autoregressive pretraining, decoder-only models can be efficiently adapted into non-causal decoders, simply by extending pretraining with span corruption. This adaptation produces a second model, which can provide excellent zero-shot generalization after multitask finetuning. Accordingly, we follow their recommendation, and train an autoregressive decoder-only model which we will later consider adapting and finetuning.

2.2 Experimental Setup

We reproduce the architecture and hyperparameters from GPT-3 (Brown et al., 2020). For learning rate, we use a maximum value of 2×10^{-4}, with a linear warm-up over 375M tokens, followed by cosine decay to a minimum value of 1×10^{-5}. We use a 1M tokens batch size, with linear ramp-up over the first 4B tokens, and a sequence length of 2,048. For optimization, we use Adam (Kingma and Ba, 2014), with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, weight decay 0.1, and gradient clipping to 1.0. We also tie the word embedding and softmax matrix (Press and Wolf, 2017). Unless noted otherwise, we conduct most experiments with 1.3B parameters models, pretraining on 112B tokens.

We picked this size and dataset size as a compromise between compute cost and the likelihood that our conclusions would transfer to the target 100B+ model. Notably, we needed to be able to reliably measure zero-shot generalization above random chance. We note that training for 112B tokens our models bring them significantly above the optimality threshold identified by Kaplan et al. (2020), guaranteeing that they won't be under-trained.

The main architectural difference with GPT-3 is that all our layers use full attention, while GPT-3 uses alternating sparse attention layers (Child et al., 2019). The main value of sparse attention layers is to save compute with long sequence lengths. However at the 100B+ scale, sparse attention layers provide negligible compute savings, as the vast majority of the compute is spent on the large feed-forward layers. Kaplan et al. (2020) estimated the amount of compute per token to be:

$$C_{\text{forward}} = 2 \times (12 n_{\text{layer}} d^2 + n_{\text{layer}} n_{\text{ctx}} d),$$

where C_{forward} is the cost for the forward pass, n_{layer} is the number of layers, d is the hidden dimension, and n_{ctx} is the sequence length. This means if $12d >> n_{\text{ctx}}$, the second $n_{\text{layer}} n_{\text{ctx}} d$ term is negligible, which is the case for our final model where $d > 10,000$ and $n_{\text{ctx}} = 2048$.

FLOPS or FLOPS? We report throughput per GPU in FLOPS and total budgets in PF-days (i.e. one PFLOPS sustained for a day). It is important to highlight that FLOPS are never directly measured, but always estimated, with widely different practices across papers. We refer to *model* FLOP the estimates based the $C = 6ND$ formula from Kaplan et al. (2020), where C is the total compute, N the model size, and D the number of tokens processed. These are the FLOP actually used to train the model, and which are used for scaling laws. We refer to *hardware* FLOP the estimates reported by our codebase, using the formula from Narayanan et al. (2021). This notably includes gradient checkpointing and a more thorough accounting of operations depending on model shape.

Model	Parameters	Pretraining tokens			
		Dataset	112B	250B	300B
OpenAI — Curie	6.7B				<u>49.28</u>
OpenAI — Babbage	1.3B				**45.30**
EleutherAI — GPT-Neo	1.3B	The Pile			42.94
Ours	13B	OSCAR			47.09
	1.3B	The Pile	**42.79**	43.12	43.46
	1.3B	C4	42.77		
	1.3B	OSCAR	41.72		

Table 1: **Pretraining datasets with diverse cross-domain high-quality data significantly improves zero-shot generalization.** Average accuracy on EAI harness (higher is better) using different pretraining corpora and comparison with baseline models. **Bold is best 1.3B model for amount of tokens seen**, <u>underline is best overall</u>.

2.3 Evaluation Benchmarks

We measure upstream performance with the language modeling loss on an held out sample of the pretraining dataset. For downstream performance, we could use zero/few-shot generalization, with or without task-specific finetuning.

Upstream performance is not always aligned with downstream performance, and they are sometimes even at odds (Tay et al., 2021). It is also not always possible to compare losses across architectures, objectives, and tokenizers. Given that we mainly care about downstream task performance, we base our evaluation on a diverse set of tasks.

Specifically, we choose to measure zero-shot generalization on this set of tasks. Few-shot and zero-shot results are strongly correlated: we found a Pearson correlation coefficient of 0.93 between zero-shot and few-shot performance across model sizes in Brown et al. (2020). We do not rely on finetuning as it is not how the main final model is likely to be used, given its size and the challenges associated with finetuning at the 100B+ scale.

We use the popular EleutherAI Language Model Evaluation Harness (EAI harness, Gao et al. (2021)), evaluating models across 27 diverse tasks that are similar to those used in Brown et al. (2020) (see Appendix A for details). Prompts used in the EAI harness reproduce as closely as possible the evaluation setup of GPT-3.

2.4 Baselines

We use GPT-Neo (Black et al., 2021), a 1.3B decoder-only auto-regressive language model trained on the Pile (Gao et al., 2020), and GPT-3 (Brown et al., 2020), which we access via the OpenAI API. We evaluate two models, Babbage and Curie. We believe Babbage is 1.3B while Curie is 6.7B based on how close our computed results are to those reported in the original paper.

3 Impact of Pretraining Data

We first study the impact of pretraining data on zero-shot generalization. More diverse pretraining data, ideally curated from a cross-domain collection of high-quality datasets, has been suggested to help with downstream task performance and zero-shot generalization (Rosset, 2020; Gao et al., 2020).

3.1 Corpora

We evaluate three possible corpora, all commonly used to train large language models:

- **OSCAR** (Ortiz Suárez et al., 2019), a multilingual, filtered version of Common Crawl;
- **C4** (Raffel et al., 2019), specifically its replication by AllenAI, a processed and filtered version of Common Crawl;
- **The Pile** (Gao et al., 2020), a diverse pretraining corpus that contains webscrapes from Common Crawl in addition to high-quality data from cross-domain sources such as academic texts and source code.

For each pretraining corpus, we train a 1.3B parameter model for 112B tokens. For the Pile specifically, motivated by good early results at 112B tokens, we train up to 300B tokens, to compare with GPT-3 models and validate against GPT-Neo.

3.2 Results

Evaluation results are outlined in Table 1. We find that training on the Pile produces models that are better at zero-shot generalization, with C4 a close second, and OSCAR significantly behind.

Importantly, this finding transfers to larger scales: as part of engineering test runs, a 13B model was trained on OSCAR for 300B tokens. We found this 13B model to underperform the 6.7B model from OpenAI API, which we attribute to the low quality of the English data in OSCAR.

We also note that our model trained on The Pile outperforms the 1.3B GPT-Neo trained on the same dataset. Finally, our 1.3B model still underperforms the 1.3B model from the OpenAI API by by 1.6%. Arguably, the most likely difference is that of data, as the GPT-3 training dataset is neither publicly available nor reproducible.

Finding 1. Diverse cross-domain pretraining data combining web crawls with curated high-quality sources significantly improves zero-shot generalization over pretraining datasets constructed from Common Crawl only.

4 Architecture Ablations

We now consider ablation studies to better identify the best positional embedding, activation function, and embedding normalization placement. We note that considerations around pretraining objectives and architecture were developed in a separate line of work summarized in section 2.1.

4.1 Positional Embeddings

One aspect of the Transformer architecture that has attracted recent significant interest is the way position information is captured within the model. Without positional embeddings, Transformers cannot order tokens against one another.

Background The Transformer paper Vaswani et al. (2017) proposed two options: static sinusoidal position embeddings and learned position embeddings (i.e., the position of each token is associated with a learned embedding vector). Learned position embeddings are popular in large language models, and are used for GPT-3. Su et al. (2021) later proposed the rotary position embedding method, where the query and key representations inside the self-attention mechanism is modified such that the attention computation captures relative distances between keys and queries. Recently, Press et al. (2022) proposed a position method which does not use embeddings, and instead directly attenuates the attention scores based on how far away the keys and queries are.

Results We compare learned, rotary, and ALiBi position embeddings, and include a baseline without position embeddings. Our results are presented in Table 2. Although learned positional embeddings outperforms rotary embeddings, ALiBi yield significant better results than all other alternatives. Surprisingly, the baseline without position information shows competitive performance. While bidirectional models require positional embeddings to determine the location of tokens, we argue autoregressive models can simply leverage the causal attention masking.

Finding 2. ALiBi positionnal embeddings significantly outperforms other embeddings for zero-shot generalization.

4.2 Activation Functions

Background. Large language models by and large still mostly use the GELU activation (Hendrycks and Gimpel, 2016). We evaluate a recently proposed alternative, SwiGLU (Shazeer, 2020), which combines both Gated Linear Units (Dauphin et al., 2016) with the Swish activation

Positional Embedding	Average EAI Results
None	41.23
Learned	41.71
Rotary	41.46
ALiBi	**43.70**

Table 2: **ALiBi significantly outperforms other embeddings for zero-shot generalization.** All models are trained on the OSCAR dataset for 112 billion tokens.

Activation function	Average EAI Results
GELU	42.79
SwiGLU	**42.95**

Table 3: **SwiGLU slightly outperforms GELU for zero-shot generalization.** Models trained on The Pile for 112 billion tokens.

function (Ramachandran et al., 2017).

With x the input of the layer, we have:

$$\text{GELU}(x) = x\Phi(x)$$
$$\text{Swish}(x) = x\sigma(x)$$
$$\text{SwiGLU}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = x_1 \text{Swish}(x_2)$$

SwiGLU uses 50% extra parameters in the feed forward layers. As suggested in Shazeer (2020), we compensate for this by reducing the hidden size of the feed forward layer.

Results. We present our results in Table 3. SwiGLU produces slightly better results than GELU; however this comes at a cost of reducing the throughput by approximately a third.

This overhead may primarily be associated with the change in the hidden size of the feedforward network. Indeed, this new size, 5,456, is divisible by neither the warp size of the GPU nor the number of streaming multiprocessors, resulting in both tile and wave quantization. Further hyperparameter tuning could yield better throughputs. In our final model, we use GELU, but recommend further benchmarks around GELU and SwiGLU.

4.3 Embedding Norm

Dettmers et al. (2021) suggests that greater stability of training can be achieved by including an extra layer normalization (Ba et al., 2016) after the embedding layer. We evaluate the performance impact of such a modification in Table 4. We note that this incurs a significant reduction in the performance of the model. However, models above 100 billion parameters are notoriously unstable and require considerable engineering efforts in order to be kept stable. If this addition provides increased stability when training, it may be valuable.

Finding 3. Adding layer normalization after the embedding layer incurs a significant penalty on zero-shot generalization.

Embedding Norm	Average EAI Results
No	**43.46**
Yes	42.24

Table 4: **Layer normalization after the embedding layer diminishes performance significantly.** Models trained on The Pile for 300 billion tokens.

5 Multilinguality

The majority of 100B+ language models have been trained exclusively on English data, with notable exceptions in Chinese (Zeng et al., 2021; Wu et al., 2021) and Korean (Kim et al., 2021). Smaller massively multilingual models have seen wider adoption (Xue et al., 2020), but these models are not suitable for zero-shot tasks. Recent zero-shot results on large GPT-like multilingual models are promising, but English-only performance is usually disappointing (Lin et al., 2021).

Training data. We train a multilingual language model in order to evaluate the effectiveness and potential impacts of this practice. We use the OSCAR dataset (Ortiz Suárez et al., 2019) for training, but here we include multiple languages, not only English as in the earlier experiments. The languages we include are Arabic, Basque, Bengali, Chinese, Catalan, English, French, Hindi, Indonesian, Portuguese, Spanish, Urdu, and Vietnamese. We sample each language with a different sampling probability that downsamples the most frequent languages and upsamples the least frequent ones, so that all languages are represented. We estimate the sampling probabilities similar to Xue et al. (2021).

English-only evaluation. We first evaluate our multilingual model on the same set of English benchmarks we have used previously, results are presented in Table 5. The multilingual model has significantly lower accuracy than the English-only model on the English benchmark, which is in line with the results from Lin et al. (2021).

Multilingual evaluation. Zero-shot multilingual evaluation is more challenging to setup because it requires writing new prompts for each new language. Therefore, instead of manually writing prompts for each language, we follow the strategy proposed by Lin et al. (2021), using English prompts for non-English examples–this can be viewed as cross-lingual zero-shot generalization.

Pretraining	Average EAI Results
English-only	**41.72**
Multilingual	38.55

Table 5: **Multilingual pretraining very significantly diminishes English zero-shot generalization.** Both models trained on OSCAR for 112B tokens.

Model	Size	EN	ZH	ES	FR	VI	AR	HI	UR	Average
XGLM (Lin et al.)	7.5B	54.5	45	38.2	50.7	47.5	47.5	43.4	42.7	46.19
XGLM (reprod.)	7.5B	53.85	45.21	41.7	49.82	47.35	46.37	43.19	42.3	46.22
XGLM	1.7B	49.68	44.63	37.39	47.94	42.75	45.65	44.35	43.19	44.45
Ours	1.3B	49.9	44.53	36.77	46.51	45.75	43.41	45.95	42.91	44.47

Table 6: **Our multilingual 1.3B model achieves accuracy on zero-shot XNLI in line with XGLM Lin et al. (2021).** First row is the reported XGLM results, and the second is our reproduction of their results to validate our multilingual evaluation setup. Last two rows show that our multilingual model matches the XGLM results.

They validated this strategy, demonstrating its ability to achieve zero-shot performance on par (sometimes even better) with human-written language-specific prompts. This is also an indicator of the cross-lingual capabilities of the model.

We evaluate on XNLI (Conneau et al., 2018), a multilingual NLI dataset that covers 8 of the languages we use for training. The task uses the following English cloze-style prompt template across all languages:

`[premise]`, right? `[MASK]`, `[hypothesis]`

The prompt fields, `[premise]` and `[hypothesis]`, are filled with the premise/hypothesis pairs in the target language. For zero-shot evaluation, the `[MASK]` token is replaced with "Yes" (for entailment), "No" (for contradiction) and "Also" (for neutral). The completion with the highest likelihood according to the model is taken as its prediction.

Our evaluation is different from the zero-shot evaluation of the XTREME benchmark (Hu et al., 2020). XTREME first finetunes the model on the English training data of each downstream task, then evaluates on the non-English dataset, attempting cross-lingual generalization. Our evaluation avoids any finetuning, and instead relies entirely on zero-shot generalization from pretraining.

Results. Table 6 shows the XNLI results of our multilingual model and how it compares to XGLM (Lin et al., 2021). We were able to reproduce the results of XGLM-7.5B which validates our evaluation setup. Furthermore, the table shows that the performance of our 1.3B is inline with the XNLI 1.7B model, validating that our multilingual setup achieves competitive results. It is worth noting that our 1.3B model is trained on only 112B tokens from 13 languages while XGLM is trained on 500B tokens from 30 languages.

6 Scaling to 176B parameters

We are now ready to scale to the final model. We have established that a curated high-quality cross-domain pretraining dataset will help boost zero-shot generalization; adopted from Anonymous (2022) that it is best to use a decoder-only model with an auto-regressive language modeling objective; identified that ALiBi positional embeddings should be used; decided that a default GELU activation should be preferred; and we have motivated the use of layer normalization on the embeddings to help with model stability. We should now determine which model architecture (e.g., number of parameters, layers, width) will result in the best model out of our compute budget.

Compute budget. We have been allocated 18 weeks of dedicated use of a cluster with 52 nodes of 8 80GB A100 GPUs. We set four nodes aside as spare, as this cluster is brand new and may experience hardware failures. This amounts to 1,161,216 A100-hours in total. Assuming a throughput of 100 model TFLOPS, approximately corresponding to state-of-the-art hardware FLOPS of 150 (Narayanan et al., 2021), we have a compute budget of 4,838 PF-days for the model training. We round this down to 4,500 PF-days, this $\sim 10\%$ safety margin accounting for potential downtime and inefficiencies (e.g., batch size ramp-up) during training. To put this number in perspective, this is $\sim 23\%$ more than the training budget of GPT-3.

6.1 Parameters, Tokens, and Shapes

Fitting scaling laws. We establish scaling laws (Kaplan et al., 2020) to verify the scaling behaviour of our model and codebase, and to help decide on optimal model size. We use English data from OSCAR (Ortiz Suárez et al., 2019) for pretraining, and train 125M, 350M, 760M, 1.3B, and 13B models for 100B to 300B tokens. Fig-

Model	Size [Bparams.]	Pretraining [Btokens]	Budget [PF-days]	Layers	Hidden dim.	Attention heads num.	dim.
LaMDA (Thoppilan et al., 2022)	137	432	4,106	64	8,192	128	64
GPT-3 (Brown et al., 2020)	175	300	3,646	96	12,288	96	128
J1-Jumbo (Lieber et al., 2021)	178	300	3,708	76	13,824	96	144
PanGu-α (Zeng et al., 2021)	207	42	604	64	16,384	128	128
Yuan (Wu et al., 2021)	245	180	3,063	76	16,384		
Gopher (Rae et al., 2021)	280	300	4,313	80	16,384	128	128
MT-530B (Smith et al., 2022)	530	270	9,938	105	20,480	128	160

Table 7: **State-of-the-art 100B+ models with publicly available details.** Compute budget is expressed in model PF-days required for training the models, from the $C = 6ND$ approximation of Kaplan et al. (2020). Number of tokens for LaMDA is inferred from reported compute budget and size. Yuan did not report attention head details.

ure 1 shows the smooth decrease in loss against the compute budget as the model size increases.

From Figure 1, we observe the following fit $L(C) = (C_c/C)^{\alpha_C}$ with $\alpha_C \approx 0.046$ and $C_c \approx 253 \times 10^8$ PF-days. This scaling exponent α_C is in-line with the 0.050 reported in Kaplan et al. (2020) and the 0.048 reported at larger scale in Henighan et al. (2020). This is a remarkably close fit given that we are using different datasets, codebases, and hyperparameters. Given how close our fits are, for the rest of this section, we are going to use the coefficients derived in Kaplan et al. (2020), as they are derived from a larger set of runs.

One caveat is that these scaling coefficients are estimated on an English setup, but we will apply them to the training of a multilingual model. Comparing the loss curves of the English and multilingual 1.3B models, we observe similar scaling and convergence trends. This suggests that the capacity of the model is the same whether it is trained on English or multilingual text. Therefore, we expect English-based scaling laws coefficients to still apply for a multilingual model.

Optimality vs. convergence. From the scaling laws (Kaplan et al., 2020), it is possible to derive a Pareto frontier describing the optimal allocation of the compute budget between model size and number of tokens seen. This optimal allocation achieves the lowest possible loss for a given compute budget. It is notable that this Pareto optimal frontier describes very large models trained on few tokens; we call this training to *optimality*. This is in stark contrast with the common practice of training much smaller models on many more tokens to *convergence*. For instance, for a 1.3B parameters model, Kaplan et al. (2020) predicts optimality at 20B tokens; any additional pretraining compute budget would be better allocated to a larger model.

However, this only provides a partial view. Scaling laws focus on the upstream performance (pretraining loss) as the main evaluation metric, assuming that it directly translates to downstream performance. However, prior work showed that this is not always the case (Tay et al., 2021). Additionally, Table 1 shows that zero-shot generalization continues improving significantly past optimality.

Scaling laws also neglect inference cost. A larger model is necessarily more expensive to serve, and inference costs at scale can rapidly catch up with training costs (Patterson et al., 2022). Finally, in the context of multilinguality , we note that most multilingual models are also trained for more tokens (e.g. XGLM on 500B tokens Lin et al. (2021)), as the pretraining dataset will otherwise include very few tokens from low-resource languages.

Accordingly, we choose to use the optimality front as an upper bound on model size and a lower bound on number of training tokens. Given our compute budget, scaling laws predict an optimal allocation for training a 392B parameter model for 165B tokens. Consequently, we will use these as constraints: the largest model we can afford is 392B parameters, and the minimum number of tokens to use is 165B tokens.

Prior models. We outline in Table 7 the architectures of all publicly detailed 100B+ parameters models. We note that most of these models are trained significantly beyond their respective optimality threshold, in the 300-400B tokens range. PanGu-α is the biggest exception, but this appears to be due to hardware availability constraints rather than actively motivated by a modeling decision.

Model shape. Kaplan et al. (2020) studied the dependence of the loss with model shape, and found only a limited impact within a wide range of feed-forward ratios d_{ff}/d_{model}, aspect ratios

Model	Size [params.]	Layers	Hidden dim.	Attention heads num.	Attention heads dim.	Memory [GB]	Performance [sec/iter.]	Performance [TFLOPs]
(1)	178	82	13,312	64	208	63	104	152
(2)	178	82		128	104	60	109	146
(3)	**176**	**70**	**14,336**	**112**	**128**	**59**	**105**	**150**

Table 8: **We choose configuration (3) as the final configuration for our 176B model.** (1) was rejected because of high attention heads dimension, and (3) was favored over (2) because of higher throughput. Appendix B details all 20 final configurations benchmarked, only the best three are displayed here.

d_{model}/n_{layer}, and attention head dimensions. Notably, for a 1.5B parameters model, aspect ratios from 10-250 and head dimensions from 15 to 250 all results in less than 1% change to the final loss.

Levine et al. (2020) proposed a theoretically motivated and empirically backed law describing the optimal compromise between width and depth. Notably, for models past 100B+ parameters, they predict that models such as GPT-3 are too deep, while models in the 10B or smaller range are usually too shallow. For a GPT-3-sized model with 175B parameters, they predict an ideal depth of 80 layers.

6.2 Final Model Architecture

From the observations above, we identify three main guidelines for our final model:

- **300-400B tokens.** We want to guarantee our model will train on around 300-400B tokens of data. This is in the upper range for models in the size range we are pursuing, ensuring that low-resource languages will not be allocated too few tokens. Using the $C = 6ND$ approximation (Kaplan et al., 2020), with $C = 4,500$ PF-days and $D =$ 300-400B tokens, this constrains the model size to be around 160-200B parameters.
- **70-80 layers.** From Levine et al. (2020) and the size constraint above, we estimate that our model should have between 70 and 80 layers.
- **Maximum throughput.** Finally, we want the final architecture to have as high of a throughput per GPU as possible, as more compute will translate directly into longer pretraining and thus a better model. Engineering constraints also come into light here: wide shallow models are typically easier to parallelize across nodes, up to a point where excessive tensor paralellism becomes necessary due to memory constraints.

From these guidelines, we benchmark 20 model configurations, detailed in Appendix B. Among these configurations, we select three of particular interest, outlined in Table 8. They best fit our guidelines above, and offer high throughput.

We discard configuration (1), as its attention heads are much larger than other models in the literature. Configuration (3) is shallower than recommended by Levine et al. (2020), but delivers 3% higher throughput compared to (2). Thus, we choose configuration (3) and its better throughput, and because a shallower model is easier to deal with at inference time by introducing less latency.

7 Conclusion

Seeking to establish the best possible model architecture that can be accommodated within a fixed 1,000,000 GPU-hours compute budget, we have presented an extensive study on principled modeling decisions for large language models.

First, we have found that complimenting Common Crawl data with high-quality cross-domain curated data can boost zero-shot generalization, validating previous suggestions (Rosset, 2020; Gao et al., 2020). Through an ablation study, we have identified ALiBi as the position embedding of choice, confirmed the potential of SwiGLU, and highlighted that stabilizing techniques such as embedding normalization sometimes come at the expense of zero-shot generalization. Exploring multilinguality, we have found that multilingual models significantly underperform their monolingual counterpart on English zero-shot benchmarks.

Finally, we further provided insights into how scaling laws can be used in practice in the design of large language models. At variance with previous works, we outlined the full reasoning behind every architectural parameters, including model shape.

We hope our work can help practitioners better understand modeling decisions, leading to better language models, and that our transparency will accelerate future similar work.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Anonymous Anonymous. 2022. What language model architecture and pretraining objective work best for zero-shot generalization?

Stéphane Aroca-Ouellette, Cory Paik, Alessandro Roncone, and Katharina Kann. 2021. PROST: Physical reasoning about objects through space and time. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4597–4608, Online. Association for Computational Linguistics.

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

Qiang Ning Ben Zhou, Daniel Khashabi and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *EMNLP*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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*, volume 33, pages 1877–1901.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: an 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

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

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *ArXiv*, abs/2003.11080.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

Matt Gardner Johannes Welbl, Nelson F. Liu. 2017. Crowdsourcing multiple choice science questions.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

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*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Boseop Kim, Hyoungseok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dong Hyung Seo, Heungsub Lee, Minyoung Jeong, Sungjae Lee, Minsub Kim, SukHyun Ko, Seokhun Kim, Taeyong Park, Jinuk Kim, Soyoung Kang, Na-Hyeon Ryu, Kang Min Yoo, Minsuk Chang, Soobin Suh, Sookyo In, Jinseong Park, Kyungduk Kim, Hiun Kim, Jisu Jeong, Yong Goo Yeo, Dong hyun Ham, Do-Hyoung Park, Min Young Lee, Jaewoo Kang, Inho Kang, Jung-Woo Ha, Woo Chul Park, and Nako Sung. 2021. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers. *ArXiv*, abs/2109.04650.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

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.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. Technical report, AI21 Labs.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Ves Stoyanov, and Xian Li. 2021. Few-shot learning with multilingual language models. *ArXiv*, abs/2112.10668.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *CoRR*, abs/2007.08124.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the*

International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–15.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim. Leibniz-Institut für Deutsche Sprache.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Hung Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeffrey Dean. 2022. The carbon footprint of machine learning training will plateau, then shrink.

Mohammad Taher Pilehvar and os'e Camacho-Collados. 2018. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

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

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Corby Rosset. 2020. Turing-nlg: A 17-billion-parameter language model by microsoft.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang A. Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M SAIFUL BARI, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Rose Biderman, Leo Gao, T. G. Owe Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *ArXiv*, abs/2110.08207.

Noam Shazeer. 2020. Glu variants improve transformer.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2021. Scale efficiently: Insights from pre-training and fine-tuning transformers. *ArXiv*, abs/2109.10686.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Shaohua Wu, Xudong Zhao, Tong Yu, Rongguo Zhang, Chong Shen, Hongli Liu, Feng Li, Hong Zhu, Jiangang Luo, Liang Xu, et al. 2021. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning. *arXiv preprint arXiv:2110.04725*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *NAACL*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. 2021. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*.

A Evaluation details

Task		Type	Random baseline
ARC (Clark et al., 2018)	Challenge	Natural Language Inference	25.0
	Easy		25.0
GLUE	MRPC (Dolan and Brockett, 2005)	Paraphrase Identification	50.0
	QQP (Iyer et al., 2017)	Paraphrase Identification	50.0
HellaSwag (Zellers et al., 2019)		Sentence Completion	25.0
LAMBADA (Paperno et al., 2016)		Sentence Completion	0.0
LogiQA (Liu et al., 2020)		Multiple-Choice Question Answering	25.0
MathQA (Amini et al., 2019)		Multiple-Choice Question Answering	20.1
MC-TACO (Ben Zhou and Roth, 2019)		Multiple-Choice Question Answering	36.2
OpenBookQA (Mihaylov et al., 2018)		Multiple-Choice Question Answering	25.0
PIQA (Bisk et al., 2020)		Multiple-Choice Question Answering	50.0
PROST (Aroca-Ouellette et al., 2021)		Multiple-Choice Question Answering	25.0
PudMedQA (Jin et al., 2019)		Multiple-Choice Question Answering	33.3
QNLI (Rajpurkar et al., 2016; Wang et al., 2019)		Sentence Completion	50.0
Race (Lai et al., 2017)		Closed-Book Question Answering	25.0
SciQ (Johannes Welbl, 2017)		Multiple-Choice Question Answering	25.0
SST (Socher et al., 2013)		Sentiment	50.0
SuperGLUE	Boolq (Clark et al., 2019)	Multiple-Choice Question Answering	50.0
	COPA (Gordon et al., 2012)	Sentence Completion	50.0
	MultiRC (Khashabi et al., 2018)	Multiple-Choice Question Answering	5.8
	RTE (Dagan et al., 2005)	Natural Language Inference	50.0
	WIC (Pilehvar and os'e Camacho-Collados, 2018)	Word Sense Disambiguation	50.0
	WSC (Levesque et al., 2012)	Word Sense Disambiguation	50.0
TriviaQA (Joshi et al., 2017)		Closed-Book Question Answering	0.0
WebQuestions (Berant et al., 2013)		Closed-Book Question Answering	0.0
Winogrande (Sakaguchi et al., 2019)		Coreference resolution	50.0
WNLI (Sakaguchi et al., 2019)		Natural Language Inference	50.0
EAI harness			**33.3**

Table 9: **Evaluation tasks considered in the EAI harness and random baselines.**

B Architecture details

ARCHITECTURE					PARALLELISM				PERFORMANCE		
Size [Bparams.]	**Hidden dim.**	**Layers**	**Attention heads** num.	**Attention heads** dim.	**Data**	**Tensor**	**Pipeline**	**MBS**	**Memory** [GB]	**Throughput** [s/iter.]	**Throughput** [TFLOPs]
206	14,336	82	128	112	8	4	12	2	OOM		
203	13,312	94	128	104	8	4	12	2	67	124,1	146,1
195	12,288	106	128	96	8	4	12	2	67	121,4	143,7
			96	128				4	79	120,3	145,0
				128				2	65	118,8	146,9
			64	192					67	116,5	149,8
184	12,288	100	64	192	16	4	6	2	OOM		
								1	OOM		
					8	8		4	72	121,0	136,2
								2	61	140,0	117,9
178	13,312	82	128	104	8	4	12	2	60	108,8	145,7
			104	128					62	123,7	128,1
			64	208	4	8		4	74	104,8	151,2
									52	111,8	141,8
					8	4		2	63	104,5	151,7
176	14,336	70	128	112	8	4	12	2	60	105,9	148,1
			112	128					59	104,5	150,1
			64	224				4	73	102,3	153,3
								2	59	102,0	153,7
					4	8	12		40	121,6	128,9

Table 10: **Throughput and memory usage of considered models sizes.** Note that pipeline parallelism here considers equal "slots" for embeddings and Transformer layers. This is important to optimize pipeline use, as our multilingual embeddings are quite large (250k).

Public Name			OpenAI: babbage	Openai: curie	gpt-neo 1.3B													
Dataset						C4	OSCAR	The Pile	The Pile	The Pile	The Pile	The Pile	OSCAR	The Pile	OSCAR	OSCAR	OSCAR	OSCAR-ML
Embeddings						Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Learned	Rotary	ALiBi	None	Learned
Activation						GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU	SwiGLU	GELU	GELU	GELU	GELU
Embedding Norm						No	No	No	No	No	No	No	No	No	No	No	No	No
Parameters in billion			1.3	6.7	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	13	1.3	1.3	1.3	1.3	1.3
Tokens trained in billion			300	300	300	112	112	112	250	300	300	330	300	112	112	112	112	112
task	metric																	
arc_challenge	acc	arc_challengeacc	0.276	0.334	0.231	0.243	0.249	0.258	0.264	0.260	0.242	0.250	0.322	0.247	0.236	0.252	0.249	0.212
arc_challenge	acc_norm	arc_challengeacc_norm	0.295	0.375	0.259	0.274	0.261	0.275	0.277	0.286	0.277	0.290	0.342	0.268	0.270	0.276	0.260	0.243
arc_easy	acc	arc_easyacc	0.597	0.685	0.562	0.561	0.560	0.556	0.569	0.601	0.568	0.582	0.681	0.557	0.554	0.575	0.537	0.484
arc_easy	acc_norm	arc_easyacc_norm	0.555	0.633	0.502	0.503	0.478	0.506	0.518	0.528	0.516	0.515	0.600	0.502	0.476	0.491	0.461	0.434
boolq	acc	boolqacc	0.629	0.666	0.620	0.546	0.566	0.520	0.551	0.606	0.558	0.566	0.587	0.540	0.584	0.563	0.526	0.597
copa	acc	copaacc	0.810	0.850	0.690	0.700	0.720	0.710	0.710	0.730	0.690	0.690	0.880	0.660	0.690	0.780	0.680	0.710
hellaswag	acc	hellaswagacc	0.429	0.504	0.387	0.422	0.404	0.374	0.385	0.405	0.378	0.380	0.542	0.379	0.410	0.422	0.395	0.340
hellaswag	acc_norm	hellaswagacc_norm	0.545	0.664	0.489	0.551	0.515	0.464	0.486	0.521	0.477	0.476	0.716	0.475	0.524	0.549	0.495	0.424
lambada	acc	lambadaacc	0.625	0.694	0.572	0.469	0.481	0.569	0.575	0.609	0.581	0.580	0.634	0.574	0.496	0.501	0.454	0.408
logiqa	acc	logiqaacc	0.201	0.215	0.197	0.206	0.237	0.210	0.218	0.203	0.217	0.223	0.232	0.215	0.210	0.215	0.237	0.218
logiqa	acc_norm	logiqaacc_norm	0.269	0.292	0.273	0.267	0.270	0.275	0.286	0.269	0.281	0.280	0.275	0.272	0.254	0.272	0.293	0.283
mathqa	acc	mathqaacc	0.244	0.251	0.241	0.233	0.222	0.249	0.248	0.263	0.246	0.245	0.238	0.245	0.234	0.237	0.215	0.223
mathqa	acc_norm	mathqaacc_norm	0.242	0.247	0.237	0.228	0.228	0.246	0.245	0.259	0.242	0.242	0.235	0.234	0.229	0.238	0.221	0.222
mc_taco	f1	mc_tacof1	0.458	0.484	0.493	0.361	0.293	0.485	0.488	0.494	0.487	0.489	0.497	0.493	0.461	0.337	0.477	0.387
mrpc	acc	mrpcacc	0.578	0.684	0.684	0.684	0.588	0.684	0.684	0.684	0.679	0.679	0.677	0.684	0.684	0.684	0.679	0.302
mrpc	f1	mrpcf1	0.718	0.812	0.812	0.812	0.702	0.812	0.812	0.812	0.808	0.809	0.806	0.812	0.812	0.812	0.808	0.090
multirc	acc	multircacc	0.018	0.015	0.018	0.018	0.026	0.023	0.024	0.023	0.025	0.008	0.018	0.026	0.009	0.011	0.016	0.040
openbookqa	acc	openbookqaacc	0.224	0.290	0.216	0.220	0.200	0.190	0.196	0.222	0.194	0.208	0.294	0.214	0.212	0.224	0.210	0.170
openbookqa	acc_norm	openbookqaacc_norm	0.336	0.386	0.336	0.336	0.328	0.316	0.314	0.334	0.302	0.312	0.412	0.320	0.344	0.340	0.332	0.276
piqa	acc	piqaacc	0.745	0.763	0.711	0.732	0.716	0.693	0.704	0.716	0.698	0.706	0.777	0.693	0.720	0.729	0.711	0.674
piqa	acc_norm	piqaacc_norm	0.746	0.772	0.711	0.730	0.721	0.705	0.705	0.717	0.698	0.701	0.788	0.689	0.721	0.731	0.711	0.682
prost	acc	prostacc	0.270	0.288	0.238	0.243	0.237	0.249	0.229	0.204	0.219	0.226	0.281	0.244	0.287	0.280	0.240	0.253
prost	acc_norm	prostacc_norm	0.260	0.295	0.308	0.293	0.303	0.268	0.271	0.268	0.292	0.305	0.283	0.276	0.296	0.332	0.300	0.313
pubmedqa	acc	pubmedqaacc	0.611	0.622	0.544	0.573	0.438	0.563	0.589	0.662	0.612	0.612	0.615	0.589	0.507	0.514	0.486	0.412
qnli	acc	qnliacc	0.512	0.529	0.499	0.476	0.507	0.505	0.506	0.505	0.499	0.499	0.517	0.498	0.493	0.481	0.493	0.493
qqp	acc	qqpacc	0.372	0.441	0.382	0.396	0.384	0.381	0.370	0.375	0.371	0.369	0.368	0.435	0.370	0.423	0.370	0.389
qqp	f1	qqpf1	0.534	0.515	0.522	0.530	0.519	0.534	0.537	0.537	0.538	0.538	0.533	0.495	0.539	0.475	0.537	0.505
race	acc	raceacc	0.356	0.386	0.341	0.330	0.323	0.334	0.329	0.344	0.321	0.323	0.374	0.337	0.317	0.344	0.332	0.326
rte	acc	rteacc	0.585	0.552	0.603	0.502	0.534	0.563	0.549	0.578	0.563	0.549	0.524	0.527	0.545	0.524	0.527	0.505
sciq	acc	sciqacc	0.867	0.919	0.860	0.825	0.810	0.838	0.853	0.868	0.860	0.867	0.895	0.849	0.818	0.828	0.816	0.793
sciq	acc_norm	sciqacc_norm	0.809	0.896	0.770	0.747	0.717	0.755	0.762	0.792	0.791	0.803	0.815	0.770	0.718	0.728	0.698	0.702
sst	acc	sstacc	0.732	0.666	0.656	0.676	0.560	0.753	0.721	0.501	0.528	0.710	0.514	0.760	0.493	0.588	0.588	0.510
triviaqa	acc	triviaqaacc	0.115	0.195	0.052	0.027	0.025	0.056	0.065	0.058	0.047	0.049	0.133	0.050	0.031	0.039	0.028	0.021
webqs	acc	webqsacc	0.048	0.065	0.017	0.012	0.004	0.023	0.026	0.023	0.020	0.021	0.027	0.012	0.006	0.004	0.015	0.001
wic	acc	wicacc	0.495	0.500	0.500	0.495	0.508	0.495	0.500	0.500	0.498	0.500	0.498	0.500	0.498	0.492	0.500	0.500
winogrande	acc	winograndeacc	0.595	0.648	0.551	0.564	0.565	0.536	0.552	0.560	0.533	0.543	0.647	0.538	0.564	0.583	0.543	0.519
wsc	acc	wscacc	0.394	0.558	0.365	0.539	0.567	0.365	0.365	0.365	0.414	0.385	0.500	0.365	0.394	0.635	0.462	0.539
Avg acc			45.30%	49.28%	42.94%	42.77%	41.72%	42.79%	43.12%	43.46%	42.24%	43.08%	47.09%	42.95%	41.45%	43.70%	41.23%	38.55%