

EXPERIMENTAL EVALUATION OF ALGORITHMIC PROGRESS IN AI

Anonymous authors

Paper under double-blind review

ABSTRACT

Algorithms have been estimated to increase AI training efficiency by a factor of 10^4 (Ho et al, 2024). We run controlled ablation experiments and benchmark the gains in training efficiency from many recent training improvements, but find that they only account for a modest 10x of these gains at small scales. We postulate that significant gains come from algorithms improving with compute scale. We run scaling law experiments, ablating important algorithmic improvements, and find evidence that some may not only affect efficiency but also the scaling laws themselves. In the process, our study illustrates the interaction between algorithmic improvements and clarify how to think about algorithmic advances in AI. Finally, we suggest revising commonly held assumptions for modeling advances in AI algorithms and rethinking the role of algorithmic advances in AI.

1 INTRODUCTION

Algorithms have played an important role in machine learning progress. (Ho et al., 2024) estimated that algorithmic progress has contributed more than 4 orders of magnitude to compute scaling in training in the last 10 years. However, their contribution is hard to estimate. In this respect, they constitute the dark energy of training - an unseen quantity that is responsible for the expansion of deep learning. Here, we try to probe deeper into the root causes of algorithms. We find that in analogy with cosmological dark energy, some algorithmic advances matter much more at larger scales. In the process, our analysis sheds light on the tenets of the dominant viewpoint of algorithmic progress (Davidson et al., 2023). In the dominant framework, algorithmic advances are seen as relatively smooth, continuous scale-independent advances that multiplicatively build on each other, leading to smooth exponential growth in effective compute over time. Ho et al. (2024) estimates that algorithmic progress increases effective compute around 3x per year. Using ablation experiments, our research calls into question many of these tenets.

The dominant framework in the algorithmic progress literature views algorithmic advances as multiplicative advances, which multiply the amount of effective compute and which don't have noticeable efficiency changes with scale Davidson et al. (2023) Ho et al. (2024). For instance, (Ho et al., 2024) estimates that the invention of the transformer led to a 60x increase in effective compute, i.e, previous architectures like LSTMs would have required 60 times the physical compute resources to match the transformer. This is based on research such as the (Hestness et al., 2017), which found no scaling exponent difference between architectures LSTMs and RHN (Recurrent Highways Networks), as well as between Adam and SGD optimizers. (Bansal et al., 2022) measured the data scaling exponent between Encoder-Decoder, Decoder only, and mixed LSTM-Transformer architectures and found little difference.

In parallel, ML theory has increasingly focused on the intrinsic properties of training data as the most important factor in determining the exponent in neural scaling laws (Michaud et al., 2023)(Bahri et al., 2024).

However, this paradigm has started to come under criticism. Sanderson et al. (2025) examines the compute multipliers from selected papers and speculates that algorithmic advances like the LSTM-Transformer transition had scale-dependent effects. There are also new AI architectures that are purported to have a larger/steeper scaling exponent (Liu et al., 2024). Finally, Droppo & Elibol (2021) studies acoustic models and finds that LSTMs scale more efficiently than transformers.

2 SCALING EXPERIMENTS DESCRIPTION

Broadly, we study two different "model" models. The first is a transformer-based model. The second is an LSTM based on the architecture and parameter. We try to keep as many things consistent as possible between the two models while adjusting some hyperparameters to keep optimal performance on each model, respectively. We base as many hyperparameter choices as possible on state-of-the-art parameters for small models in the literature. Sometimes we have to choose between different recommendations in the literature or choose different parameters to keep consistency between models. In these cases, we run variation experiments to justify our choices (see Appendix 9). We also run learning rate tunes for all our models at every scale.

Our experimental setup consists of a transformer and an LSTM-based model. We attempt to control for as many confounders as possible while using hyperparameters that are consistent with the respective architecture. All the models in this paper share the same tokenizer and use the same selection of the C4 dataset. Our architectural and hyperparameter choices for our transformer model are primarily based on those in Srećković et al. (2025), Hoffmann et al. (2022), and Porian et al. (2024). For parameters not specified in these papers, we do a small tuning study or find other relevant literature to base our parameters on. Our LSTM model is based on Melis et al. (2017), where we use the parameters from their word-level 10M parameter model. We do learning rate tunes across all our models at all scales. See Appendix 8.1 and the following discussion for more details on our experimental setup.

2.1 TRANSFORMER MODEL

For our transformer-based experiments, we use a vanilla transformer with rotary-based encodings. We keep a constant width to depth aspect ratio. Kaplan et al. (2020) used a $N : L \approx 100$ width-depth ratio (Dey et al., 2025). At the scale of our experiments, we choose a ratio of $N : L \approx 16$. This gives us depth to width ratio similar to that of other small transformers like Lan et al. (2019). Attention heads number is commonly set such that the dimension of each head stays at a constant size, such as 64 or 128 (Hoffmann et al., 2022)(Vaswani et al., 2017). Given the scale of our experiments, we choose a constant attention head dimension of 16. In our experience, setting a larger head dimension and therefore reducing the number of attention heads degraded performance at small scales. To scale our transformers, run a learning rate tune at each model size over the range: $[10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}]$. Each learning rate tune consist of a small training run with a fixed token budget and fixed schedule (across sizes). We measure FLOPs by running PyTorch profiler over one minibatch and scaling according to gradient accumulation and number of steps. Our initialization, unless stated otherwise, has 0 bias, all weights are sampled from $\mathcal{N}(0, 0.02)$. We scale the output projections initialization by $1/\sqrt{2L}$ where L is the number of layers as is done in Wang et al. (2024) as this is more suitable for the deeper networks in our study (Radford et al., 2019). We use pre-layer normalization (Xiong et al., 2020) and experiment with RMSNorm but find it has little effect on final loss or compute efficiency (see Section 3.5).

2.2 LSTM MODEL

For our LSTM model we choose a vanilla LSTM and do not analyze more recent variations developed after 2018 (see (Melis et al., 2019) and (Beck et al., 2024)). In addition, we do not include the many pre-2018 variations of LSTMs like GRUs as Greff et al. (2016) finds that these variants do not improve on the standard LSTM. We develop a setup based on state-of-the-art LSTMs before the invention of the Transformer using hyperparameters from Melis et al. (2017). We focus our efforts on a 2-layer LSTM, which seems to perform well in practice for many studies at many sizes (Reimers & Gurevych, 2017)(Melis et al., 2017). The layer number seems to scale very moderately with LSTM size (Melis et al., 2017) so we keep the layer number constant while scaling hidden dimension for the range of scale we examine. In addition to standard LSTM optimizations ie, gradient clipping, and TBPTT (Truncated Back Propagation Through Time). All other parameters we keep constant between LSTM and Transformer (see Appendix). Similar, to our transformer model, for each scale we run a learning rate tune over: $[10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}]$. We use standard LSTM initialization consisting of Xavier-uniform embedding weights, orthogonal recurrent weights, and 0 bias except for a forget gate bias of +1. We include a variational dropout with separate dropout rates for embedding matrices vs hidden matrices (Melis et al., 2017)(Merity et al., 2017). Except for the

choice of initialization, the implementation we most closely follow (Melis et al., 2017) does not use any algorithms that were not present in 2012 at the beginning of the analysis of Ho et al. (2024).

2.3 DISCUSSION

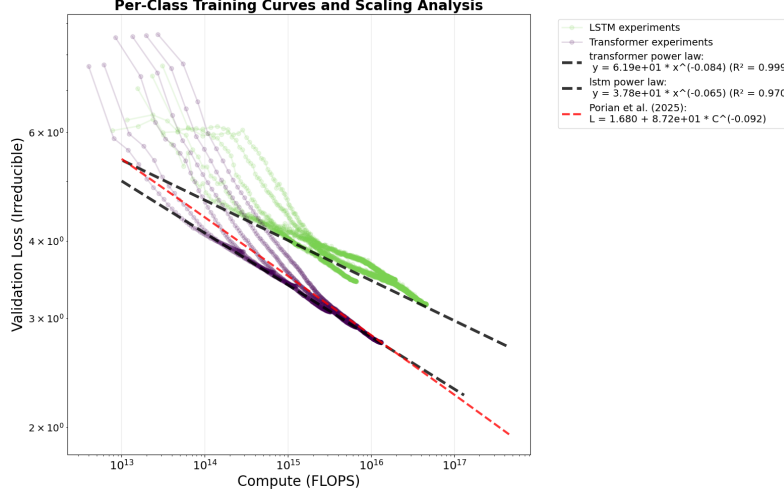


Figure 1: Graph of standard transformer vs LSTM training runs, subtracting irreducible loss. We also plot the scaling fit found by Porian et al. (2024) as a comparison to other chinchilla-based scaling studies at a small scale. LSTMs appear to have a noticeably different scaling exponent.

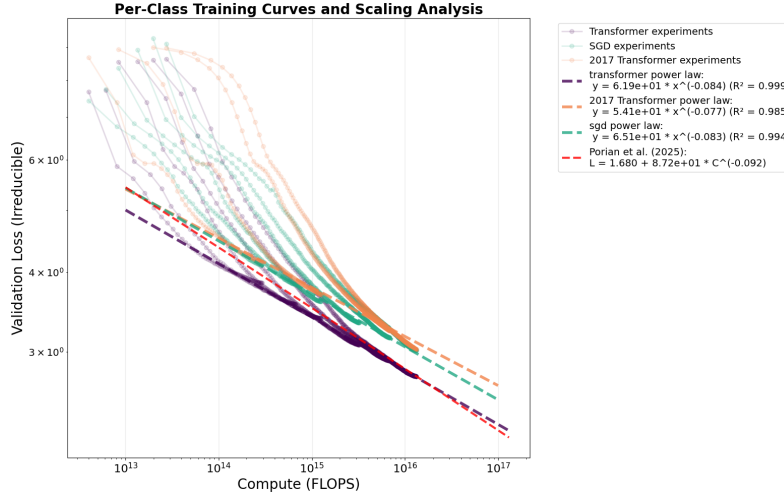


Figure 2: Graph of several training runs for a transformer with our default setting, a transformer trained with SGD (in blue), and a transformer using 2017 algorithms. We also plot the scaling fit found by Porian et al. (2024) as a comparison to other chinchilla-based scaling studies at a small scale. transformers, we see that scaling laws exponents are largely similar.

Our scaling graphs hint that improvements in neural network architecture are not scale invariant and have increasing returns to scale effects. However, optimizer choices seem to be scale-invariant. Comparing our fit with those in Porian et al. (2024) we see that our transformer performance is in-line with papers running experiments at comparable scales. Interestingly, our LSTM model at a small scale is only 2-4 times less compute-efficient than our Transformer model. Further, the gap between a model trained with adam vs sgd is less than 2 (at batch size 64). If we extrapolate the different scale trends, we see that a transformer trained with 10^{18} is comparable in performance to

an LSTM trained with $3 \cdot 10^{19}$ Flops. This is directly in line with previous research, which estimates efficiency gains of 20-50x at these scales (Sanderson et al., 2025).

3 WHAT HAS BEEN THE IMPACT OF PARTICULAR ALGORITHMIC INNOVATIONS?

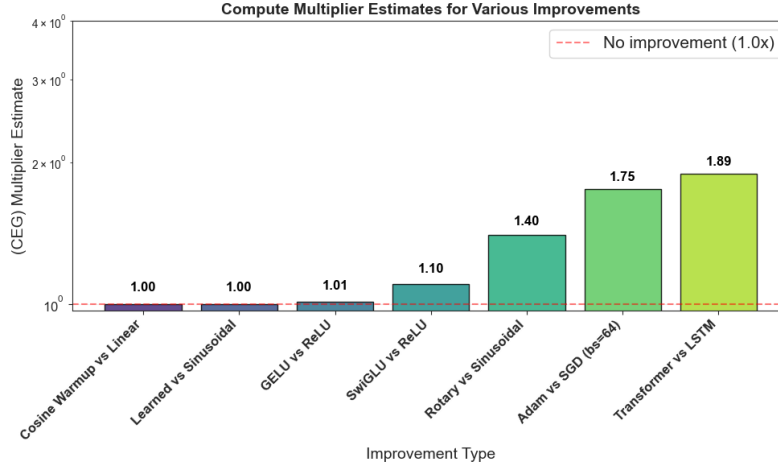


Figure 3: Compute Equivalent Gains/training efficiency gains for some of the algorithms measured in this study on a 3.6M parameter transformer model. Overall, many recent training advancements have a small impact on training efficiency. However, ML architectural changes have a significant effect.

Here, we try to get a fine-grained picture of algorithmic improvements in training by running a large series of ablation experiments. For instance, running a transformer with older activation functions like ReLU and with newer activation functions like SwiGLU. In each case, we run learning rate tunes where we sweep learning rates $10^{0.5}$ higher and lower than the learning rate of the model without the ablation. For each model, we run training with four different seeds to better identify the extent of the algorithmic improvement relative to noise. In order to run experiments with many seeds, all of these experiments are done with a small 3.6 million parameter transformer model (SwiGLU activation adds a small number of additional parameters).

We choose to examine activation functions, positional encodings, normalization techniques, learning rate schedules, initialization techniques, and optimizers. We leave out tokenizers and data quality enhancements as they are difficult to evaluate using perplexity. We examine literature that has benchmarked these contributions. However, in general, we find that estimating algorithmic improvements from the primary literature (i.e, from the papers that proposed the algorithmic improvement itself) is significantly higher than the secondary literature (i.e, estimates from papers that include multiple previous algorithms as benchmarks). The primary literature may have a bias to enlarge their innovation to make it seem more favorable. Experimentally, we find that in many of these categories (except initialization and normalization), performance has improved by 10% – 20% since the invention of the transformers at the scales we measure. We describe the effects of each of these innovations below and in Fig 3.

We implement our analysis with the widely used compute equivalent gains (CEG) framework (Davidson et al., 2023) (Sanderson et al., 2025) (Ho et al., 2024) to evaluate the effectiveness of training improvements. This is similar to Hernandez & Brown (2020), who measured the differences in compute level needed to get a fixed level of capability. However, such a measure is very sensitive to the benchmark threshold chosen (Erdil & Besiroglu, 2022). For instance, if different models plateau at different threshold, than theoretically the model with the lower plateau would require “infinitely” more compute to reach the performance of the lower model. We therefore choose the lowest loss threshold that all models are capable of because. Given that we scale only to chinchilla optimality, this means that we are comparing modles in a stable power law regime.

3.1 ACTIVATION FUNCTIONS

Interestingly, we find no statistically significant difference between GELU and ReLU activation functions. This replicates [Shazeer \(2020\)](#). In addition, [Mirzadeh et al. \(2023\)](#) finds that ReLU has a negligible impact on performance and recommends the reintroduction of ReLU into language models.

3.2 POSITIONAL ENCODING

We find that positional encodings might have some impact on training compute efficiency. We test three different types of positional encoding, including learned positional encoding. We estimate that rotary encoding constitutes an improvement of 12% in training efficiency over sinusoidal encoding originally employed in [Vaswani et al. \(2017\)](#). This is in contrast to [Sanderson et al. \(2025\)](#), which estimates a 70% gain and [Biderman et al. \(2021\)](#), which sees a 10% – 30% training efficiency gain. However, we think such efficiency gains might be highly size and sequence-dependent as rotary encodings were originally developed to incorporate larger and more flexible sequence lengths rather than as a general training efficiency improvement.

3.3 LEARNING RATE SCHEDULES

Our test consists of 3 prominent learning rate schedules used in the last 10 years. These include an inverse square root learning rate decay schedule variations of which were used in T5 [Raffel et al. \(2020\)](#) and [Vaswani et al. \(2017\)](#). We also implement a linear warmup/linear decay schedule as used in BERT ([Devlin et al., 2019](#)) and linear-warmup with cosine decay, which is a more recent default use in GPT-3 [Brown et al. \(2020\)](#). We measure some training efficiency gains due to learning rate schedulers. We observe a CEG gain of about 12% training efficiency gain between the cosine decay schedule and the inverse square root decay schedule, but notice little difference as long as the decay is faster than square root, which is the more important distinction ([Zhang et al., 2025](#)). Interestingly, [Porian et al. \(2024\)](#) finds that the cosine learning rate scheduler matter more with scale. Our results are roughly in line with [Kaplan et al. \(2020\)](#), which find minimal difference between state-of-the-art learning rate schedulers. However, [Bergsma et al. \(2025\)](#) finds that their learning rate scheduler has a 60% compute efficiency gain for 610M parameter model.

3.4 INITIALIZATION

We compare three different types of initialization. Xavier-uniform, Kaiming-normal, and transformed scaled initializations developed by [Wang et al. \(2024\)](#), which is our default. [Wang et al. \(2024\)](#) modifies the initialization developed by BERT by scaling output projections by $1/\sqrt{2L}$. In our transformer model, we find that these variations have a negligible impact on transformer pre-training efficiency. Papers like [Bachlechner et al. \(2021\)](#) developed an initialization technique which they say improves pretraining efficiency by 56% for a 12-layer transformer. However, much of the initialization literature we encountered concentrated on stabilizing training or tailoring training for specific architectures rather than broad efficiency improvement per-se.

3.5 NORMALIZATION TECHNIQUES

We test 3 different normalization techniques: pre-layer norm, post-layernorm, and rmsnorm. Pre-layernorm applies layernorm before multihead attention in the residual stream. This is generally seen as more stable than its predecessor post-layernorm. RMSNorm is an update to layernorm, which may have runtime benefits of 7% – 64% because it removes the mean centering step ([Zhang & Sennrich, 2019](#)). However, in these variations seem to have little impact on the computational efficiency of training.

3.6 OPTIMIZER ADAM VS SGD

Sgd performs notably badly on transformers in comparison to Adam ([Ahn et al., 2023](#)) ([Zhao et al., 2024](#)). This is theorized to be due to Hessian heterogeneity [Zhang et al. \(2024\)](#). In contrast, [Hestness et al. \(2017\)](#) found that Adam and SGD scale similarly with respect to data on recurrent highway networks. More recent work finds that the Adam SGD gap is very small when high momentum

is used and batch sizes are very small [Srećković et al. \(2025\)](#). Further, [Srećković et al. \(2025\)](#) argues that the optimal batch size for sgd is 1 and that at this scale there would be a negligible gap between the two optimizers. In addition, [Zhang et al. \(2019\)](#) finds that Adam is compatible with much higher critical batch sizes than SGD. At batch sizes above 128 we find it nearly impossible to get any scaling behavior from a transformer trained on sgd. However, with our default batch size of 64 along [Srećković et al. \(2025\)](#) recommended .98 momentum, we find relatively smooth scaling behavior and a relatively moderate 2-3x gap in training efficiency.

3.7 DATASET AND TOKENIZERS

We do not do any tokenizer and dataset ablation experiments. In general, it is hard to compare performance between tokenizers and datasets as perplexity is a tokenizer and dataset-relative benchmark. However, there has been significant benchmark-based efficiency studies. [Li et al. \(2024\)](#) did a large-scale study comparing pretraining efficiency on new and historical datasets. They found that their optimal dataset was able to improve pre-training efficiency by a factor of 2-3 over the worst-performing historical dataset (C4). Similarly, optimized datasets like ([Xie et al., 2023](#)) have shown pretraining efficiency gains of 2.6x. In regards to tokenizer, [Ali et al. \(2024\)](#) finds that inefficient tokenization leads to additional training costs of 68%.

3.8 KAPLAN TO CHINCHILLA SCALING LAWS

[Kaplan et al. \(2020\)](#) advocates for a smaller level of data scaling relative to parameters as compute scales relative to the chinchilla scaling procedure developed by [Hoffmann et al. \(2022\)](#). Subsequent literature found that Kaplan misapproximated scaling law exponents due to small sizes, not accounting for embedding parameters, and using a constant warmup size ([Porian et al., 2024](#)) [Pearce & Song \(2024\)](#). [Hoffmann et al. \(2022\)](#) accounted for all parameters and found that data and parameters should be scaled equally while scaling compute. We can estimate the compute efficiency gains of the switch from Kaplan to Chinchilla using the recommended data and compute allocations from each source. Here, we assume that computer scaling laws are governed by a chinchilla optimal form in reality, while model trainers allocate parameters and data according to mistaken Kaplan scaling laws (see Appendix 11). This form of algorithmic improvement has clear scale-dependent effects but has an interesting form where the training efficiency gap first converges, then diverges. Further, the training efficiency gains are large but less than 3 for much of the scale used in contemporary machine learning (ie. $10^{16} - 10^{25}$ FLOP).

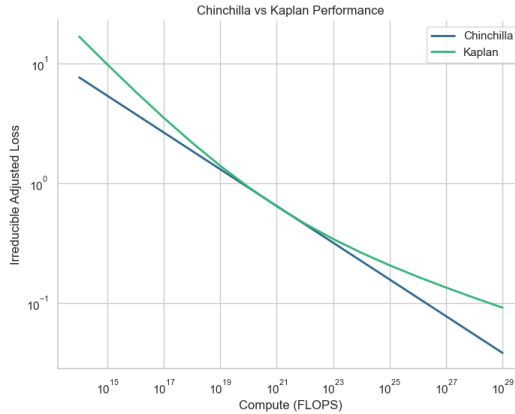


Figure 4: Difference in performance between transformer models scaled with Kaplan vs Chinchilla recommendations. Interestingly, the compute gap first converges then diverges.

3.9 INTERACTION BETWEEN ALGORITHMS

Conventional theory of algorithmic gains presupposes that most algorithmic gains are multiplicative [Ho et al. \(2024\)](#) [Davidson et al. \(2023\)](#). For instance, if algorithms A increasing training efficiency

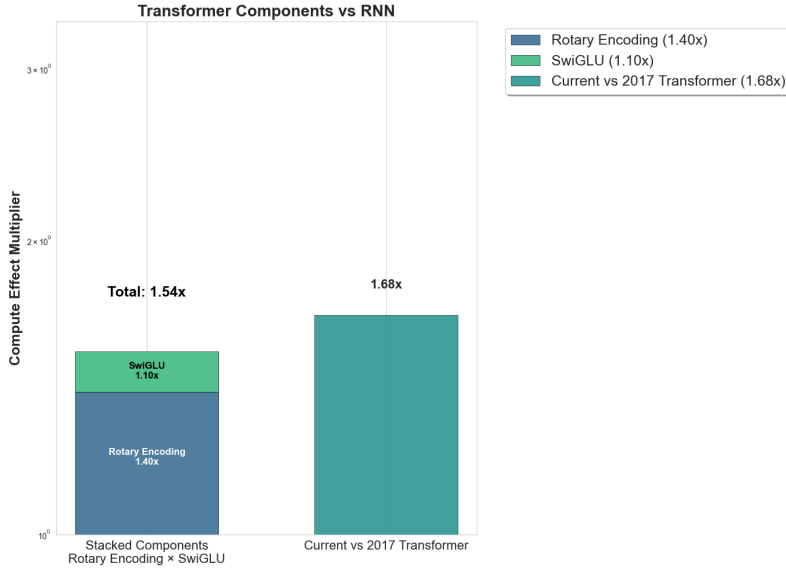


Figure 5: How well do the effects of individual algorithmic improvements account for overall algorithmic progress? We compare the overall efficiency of a transformer with recent improvements like rotary-encoding, post-layer norm, swi-glu activation functions vs the predicted effect from the analysis in Fig 3. Interestingly, it looks like the combined effect is greater than the sum of its parts.

by 2x and algorithm B increases training efficiency by 3x, respectively, then the combination of algorithm A and B should have a training efficiency gain of 6x. However, across architectures this is not the case. For example, Adam provides a compute efficiency gain for Transformers while most state of the art LSTMs used RMSprop or plain SGD over Adam (Merity et al., 2017) (Melis et al., 2017) (Keskar & Socher, 2017).

To help identify these interactions, we compare a transformer trained similar to the original transformer developed by Vaswani et al. (2017) where we remove 4 algorithmic improvements. We go from SwiGLU to GeLU, prelayernorm to postlayernorm, rotary encoding to positional encodings, and use an inverse square root learning rate schedule rather than cosine decay learning rate schedule. Using the estimates in Fig 3 and assuming multiplicative improvement we would expect CEG gains of 54%. Yet running the experiment, we find compute equivalent gains of 68%. A possible explanation for this discrepancy is that there are more hyperparameters than the learning rate that need to be tuned with such a drastic change in model configuration. However, most of our hyperparameter are shared between Vaswani et al. (2017) and Hoffmann et al. (2022).

4 LIMITATIONS AND FURTHER WORK

We would like to reiterate the limitations of our current study and encourage further work in this field. First, we only address innovations that improve FLOP efficiency. Many of the key innovations in recent years have been innovations that improve speed and GPU utilization, like Flash Attention. This is also not addressed in Ho et al. (2024) and much of the algorithmic progress literature, which leaves out an important element of progress. Further, algorithmic improvements are heavily multidimensional and examining only efficiency gains gives a warped picture of the true benefits of algorithmic innovations.

In addition, our experiments are conducted at small scales compared to more recent scaling studies (i.e Hoffmann et al. (2022)). However, our studies are done at similar scales to prominent studies like Kaplan et al. (2020) and at similar or larger scales to the papers that introduced many of these innovations Melis et al. (2017).

Another key limitation is the fickle nature of hyperparameters. For instance, we found it nearly impossible to scale a transformer trained with sgd with a batch size over 128. We are more confident

in our results because our performance aligns well with other studies like [Porian et al. \(2024\)](#) and [Melis et al. \(2017\)](#). However, our study leaves open the possibility of a "golden" hyperparameter that could significantly lower the gap between the algorithmic improvements we measure or even change the experimental scaling relationship. For instance, SGD was long thought to be inherently inferior to Adam for Transformer training, but studies like [Srećković et al. \(2025\)](#) demonstrate a very small performance gap with the right hyperparameters. We found very little literature on scaling LSTMs. Therefore, we suspect that we could find improved scaling for LSTMs if extensive hyperparameter sweeps were done at all scales. However, this becomes computationally infeasible with much larger models. Further, determining optimal scaling for transformers took many years of scientific progress. Finding such an optimal scaling for new architectures might take a similar time. In addition, hyperparameters in a general sense count as algorithmic progress/training efficiency improvements and finding more optimal parameters than were available in the past would skew views of historical algorithm improvements.

4.1 OVERALL ACCOUNTING FOR ALGORITHMIC PROGRESS

Despite, many technical advances, algorithmic gains in training efficiency look relatively modest at small scales. With the greatest gain coming from transformers (around 3x), most other innovations amount to a comparable 3x efficiency gain, and the shift from Kaplan to chinchilla theoretically amounts to another gain of 3x. This leaves a gap of around 10^2 unaccounted for between our estimates and [Ho et al. \(2024\)](#). However, the existence of scale invariant algorithmic progress makes such an analysis nonsensical. Algorithmic progress is only well defined relative to a given computational scale. At a scale of GPT-4 scale of 10^{25} ([Rahman et al., 2025](#)) Flops, it looks reasonable that algorithmic efficiency gains could be on the order of 10^4 .

5 CONCLUSION

5.1 IMPLICATIONS FOR AI PROGRESS

Many leading models of the future of AI growth take into account a large increase in training compute efficiency over time ([Kokotajlo et al., 2025](#)) ([Erdil et al., 2025](#)). However, most efficiency gains are relatively modest and, without continued compute scaling would be smaller still. Further, scale-dependent innovations mean it may be harder to identify the true impact of a given algorithmic innovation until it is scaled up massively. Limits to compute scaling, therefore, pose a considerable obstacle to improvements in training efficiency.

5.2 THE NATURE OF ALGORITHMIC PROGRESS IN AI

Overall, algorithmic improvements seem relatively modest, especially since the invention of the Transformer. In addition, progress in training efficiency at constant scale looks much more disjoint than previously thought. Very few innovations have a disproportionate effect, and year by year, the capability can increase dramatically. However, the majority of AI algorithmic innovations do not seem to be well captured by scalar improvements in training efficiency. This is particularly important in the context of the huge scale-up of computational resources in recent years. Even a very moderate scale-dependent efficiency increase can result in huge changes. Many of the improvements we tested had little effect on training efficiency but made training more flexible and stable (for instance, post-layernorm, layer-dependent initialization, and rotary encodings). There are also technical improvements that make it possible to scale in the first place, including but not limited to the invention of pipeline and tensor parallelism ([Barnett, 2025](#)). Further, some of the most important technical improvements in machine learning are not efficiency increases but techniques that unlock qualitatively different abilities (for instance, instruct fine-tuning ([Wei et al., 2021](#))). In analogy to transportation, AI algorithmic improvements do not seem to be fixed increases or speed in fuel efficiency, but instead new vehicles and new roads to travel on.

6 LLM USAGE

LLMs played a significant role in generating code for many of our experiments. All code was validated by the researchers in this paper. We also used LLMs to find relevant literature.

7 REPRODUCIBILITY STATEMENT

All code used in model training and model analysis is hosted here, along with setup instructions: https://anonymous.4open.science/r/Experimental_Progress-02EF/. The main body of our experiments section (see Sec 2) contains details on our training and analysis procedure. A detailed list of hyperparameters used is in Appendix 8.1.

REFERENCES

- Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization). *arXiv preprint arXiv:2310.01082*, 2023.
- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, et al. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3907–3924, 2024.
- Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.
- Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27), June 2024. ISSN 1091-6490. doi: 10.1073/pnas.2311878121. URL <http://dx.doi.org/10.1073/pnas.2311878121>.
- Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. Data scaling laws in nmt: The effect of noise and architecture. In *International Conference on Machine Learning*, pp. 1466–1482. PMLR, 2022.
- Peter Barnett. Compute requirements for algorithmic innovation in frontier ai models. *arXiv preprint arXiv:2507.10618*, 2025.
- Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.
- Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Straight to zero: Why linearly decaying the learning rate to zero works best for llms, 2025. URL <https://arxiv.org/abs/2502.15938>.
- Stella Biderman, Sid Black, Charles Foster, Leo Gao, Eric Hallahan, Horace He, Ben Wang, and Phil Wang. Rotary embeddings: A relative revolution. blog.eleuther.ai/, 2021. [Online; accessed].
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Tom Davidson, Jean-Stanislas Denain, Pablo Villalobos, and Guillem Bas. Ai capabilities can be significantly improved without expensive retraining, 2023. URL <https://arxiv.org/abs/2312.07413>.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, 2019.
- Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don’t be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.
- Jasha Droppo and Oguz Elibol. Scaling laws for acoustic models. *arXiv preprint arXiv:2106.09488*, 2021.
- Ege Erdil and Tamay Besiroglu. Algorithmic progress in computer vision. *arXiv preprint arXiv:2212.05153*, 2022.
- Ege Erdil, Andrei Potlogea, Tamay Besiroglu, Edu Roldan, Anson Ho, Jaime Sevilla, Matthew Barnett, Matej Vrzla, and Robert Sandler. Gate: An integrated assessment model for ai automation. *arXiv preprint arXiv:2503.04941*, 2025.
- Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016.
- Danny Hernandez and Tom B Brown. Measuring the algorithmic efficiency of neural networks. *arXiv preprint arXiv:2005.04305*, 2020.
- Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.
- Anson Ho, Tamay Besiroglu, Ege Erdil, David Owen, Robi Rahman, Zifan Carl Guo, David Atkinson, Neil Thompson, and Jaime Sevilla. Algorithmic progress in language models, 2024. URL <https://arxiv.org/abs/2403.05812>.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Henigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL <https://arxiv.org/abs/2203.15556>.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL <https://arxiv.org/abs/2001.08361>.
- Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.
- Daniel Kokotajlo, Scott Alexander, Thomas Larsen, Eli Lifland, and Romeo Dean. AI2027: A scenario forecasting the impact of superhuman ai over the next decade. Website, 2025. URL <https://ai-2027.com/>. Published April 3, 2025.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.
- Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

- Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.
- Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier lstm. *arXiv preprint arXiv:1909.01792*, 2019.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.
- Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36:28699–28722, 2023.
- Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023.
- Tim Pearce and Jinyeop Song. Reconciling kaplan and chinchilla scaling laws. *arXiv preprint arXiv:2406.12907*, 2024.
- Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *Advances in Neural Information Processing Systems*, 37:100535–100570, 2024.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- Robi Rahman, Lovis Heindrich, David Owen, and Luke Emberson. Over 30 ai models have been trained at the scale of gpt-4, 2025. URL <https://epoch.ai/data-insights/models-over-1e25-flop>. Accessed: 2025-09-22.
- Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks, 2017. URL <https://arxiv.org/abs/1707.06799>.
- Jack Sanderson, Teddy Foley, Spencer Guo, Anqi Qu, and Henry Josephson. Rethinking llm advancement: Compute-dependent and independent paths to progress. *arXiv preprint arXiv:2505.04075*, 2025.
- Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- Teodora Srećković, Jonas Geiping, and Antonio Orvieto. Is your batch size the problem? revisiting the adam-sgd gap in language modeling. *arXiv preprint arXiv:2506.12543*, 2025.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.
- Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.
- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Learning rate scheduling. https://d2l.ai/chapter_optimization/lr-scheduler.html, 2025. “Dive into Deep Learning” (version 1.0.3), accessed 20 September 2025.
- Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.
- Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.
- Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhiquan Luo. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823, 2024.
- Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024.

8 APPENDIX

8.1 MODEL ARCHITECTURE

Table 1: Base hyperparameters used to train the Transformer Model

Hyperparameter	Value
Dropout	0.0
Sequence length	128
Stride	128
Batch size	128
Gradient Clipping	1.0
Weight decay	0.01
Width/Depth Aspect Ratio	16
Learning Rate Schedule	cosine annealing
min learning rate	0.1x max lr
warmup fraction	10%
Initialization	Transformer Scaled
Activation Function	GeLU
SGD momentum	0.9
Default Optimizer	AdamW
Embedding Weight Tying	True
Tokenizer	GPT-2 Tokenizer
Vocabulary	50257
Dataset	C4
Validation Set Size	500k tokens

8.1.1 HYPERPARAMETER TUNING

We tune the learning rate for the standard transformer, transformer with sgd, standard lstm, and lstm with sgd. We used a standard selection $[10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}]$. We experimented with a wider range of learning rates but learning rates below 10^{-4} performed significantly worse.

Table 2: Base hyperparameters used to train LSTM Model

Hyperparameter	Value
Number of layers	2
TBPTT length	64
TBPTT stride	64
Batch size	128
input dropout	0.2
hidden dropout	0.1
output dropout	0.2
learning Rate Schedule	cosine annealing
min learning rate	0.1x max lr
warmup fraction	10%
Weight decay	10^{-4}
Gradient clipping	1.0
Initialization	LSTM (Orthogonal Hidden Matrices)
SGD momentum	0.9
Embedding Weight Tying	True
Tokenizer	GPT-2 Tokenizer
Vocabulary	50257
Dataset	C4
Validation Set Size	500k tokens

8.1.2 EXTENDED MODEL DISCUSSION

We use standard variational dropout for the LSTM. We set dropout equal to 0 as was done in Chinchilla scaling laws (Hoffmann et al., 2022). We experimented with different learning rate schedules. However, in our experience they had little effect on either the standard transformer, or the transformer with sgd. This is in line with Kaplan et al. (2020).

8.2 DATA SELECTION AND LOADING

We choose to the C4 dataset for both the transformer and LSTM. Hoffmann et al. (2022) found similar scaling exponents on their full dataset and on exclusively C4.

8.3 HOW CHINCHILLA SCALING LAWS CHANGE WITH ALGORITHM ADVANTAGES

9 ABLATION AND VARIATION STUDIES

Throughout this investigation, we have had to compromise between using the very best setups for any given model and keeping a fair comparison between models. Here, we have a table of ablation experiments at small scales to justify the choices we have made. For the LSTM with dropout, we use the same settings as our default LSTM, however with input dropout 0.6, output dropout 0.7, and. All experiments are done on hans: 32d model.

Table 3: compute exponents

Change	Loss Before	Loss After
	-.3421	0.434
LSTM with Adam	-0.036	.861
LSTM with Dropout	-.148	0.1858

10 EXTENDED VARIATION EXPERIMENTS

We did not include any experiments in the main paper which had less than 5% gain as that is generally below the margin of significance. Here we include the efficiency multipliers for all experiments.

11 KAPLAN-STYLE SCALING WHEN THE TRUE FRONTIER IS CHINCHILLA-OPTIMAL

In figure 4, we show loss curves from using Kaplan and Chinchilla’s recommendations for compute optimal scaling (specifically optimal scaling by analytically solving the full parameter model). We take the Chinchilla functional form for model loss $L(N, D)$ as ground-truth, only modifying the allocation of parameters and dataset size between each curve. Unsurprisingly, Chinchilla scaling is a perfect power law.

We use the constants $E = 1.69$, $A = 406.4$, $B = 410.7$, $\alpha = .34$, and $\beta = .28$ and include the functional form below for the readers’ convenience.

$$L(N, D) = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E \quad (1)$$

For the compute allocation, Chinchilla scaling explicitly defines optimal scaling of N and D to be

$$N_{\text{Chinchilla}}(C) = G\left(\frac{C}{6}\right)^{\beta/(\alpha+\beta)} \quad (2)$$

$$D_{\text{Chinchilla}}(C) = \left(\frac{1}{G}\right)\left(\frac{C}{6}\right)^{\alpha/(\alpha+\beta)} \quad (3)$$

where $G = \left(\frac{\alpha A}{\beta B}\right)^{1/(\alpha+\beta)}$. Finally, we infer the full Kaplan scaling recommendation: Figure 14 gives a functional form for compute optimal dataset scaling, and using the relationship $C = 6ND$, we get the following:

$$N_{\text{Kaplan}}(C) = (3.6 \times 10^{-6})(C)^{0.73} \quad (4)$$

$$D_{\text{Kaplan}}(C) = \frac{C^{0.27}}{(6)(3.6 \times 10^{-6})} \quad (5)$$

Finally, this gives