Epiplexity Guided Data Selection and Generation for Out-of-Distribution Generalization

Ellen Su[*1] **Andres Potapczynski**[*1] **Shikai Qiu**[1] **Edward Hughes**[2]
Andrew Gordon Wilson[1,3]

[1]New York University [2]Inherent [3]Perplexity AI
[*]Equal contribution

Abstract

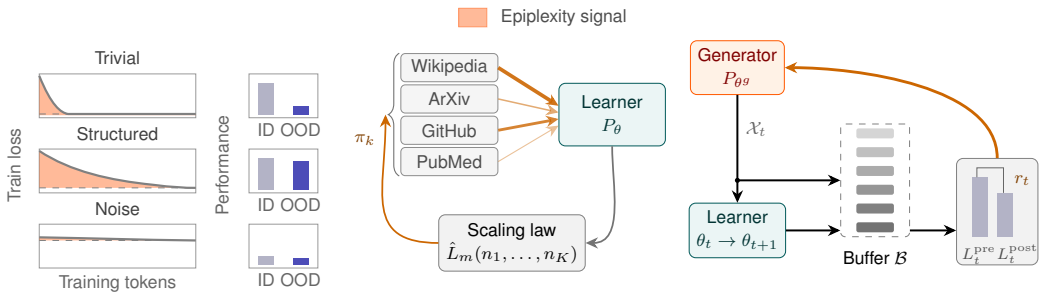
Modern systems are increasingly expected to transfer across tasks not specified during training. What data facilitates generalization to unanticipated settings? One hypothesis is that data with more structural information could contain shared circuits and subprograms that may be recycled in a wider array of downstream settings. Epiplexity, a recently proposed measure of the structural information a compute-bounded learner can extract from data, offers a mechanism to reason about this relationship. As existing estimators for epiplexity only operate on data already seen during training, we show how to operationalize epiplexity as an online training signal by greedily maximizing its stepwise gain in two settings. For data selection, we fit scaling laws to the training loss curves of natural data domains to predict the expected epiplexity gain as a function of training tokens, and use this signal to adaptively determine the sampling weights over domains during training. For synthetic data generation, where no domain structure exists, we reward a generator with the change in epiplexity of a learner's synthetic training data and use REINFORCE policy gradients to guide the generator toward an epiplexity-maximizing distribution. In both cases, we find higher epiplexity predicts improved downstream performance on zero-shot and fine-tuning tasks, supporting the hypothesis that data rich in structural information yield representations that better transfer across domains.

1 Introduction

Of model, optimizer, and data, it is *data* that is increasingly being recognized as the greatest lever for improving performance. This realization is compounded by an impending sense that we are entering a data-limited regime [1, 2], where multi-epoch training is necessary at scale. Accordingly, there is a growing body of work exploring how training data interventions influence model performance. Current methods in data mixing and selection [3–10], curriculum and ordering [11, 12], and self-play [13–19] all share the same implicit assumption that batches of data vary in how important they are to a learner, and that exploiting these differences can yield a model with better generalization. Many of these methods consider data quantity, diversity, and quality, but these notions are often vaguely defined. Moreover, models are increasingly deployed on new tasks and domains, many of which couldn't have been anticipated during training, making it even harder to reason about what properties of data ought to be desirable. In particular, how should we select data for greater *generality* in out-of-distribution (OOD) settings?

A recent theoretical development provides a promising starting point to reason about this question. Finzi et al. [20] introduced *epiplexity*, a measure of the structural information content in data, characterized as the information in the model that minimizes the description length of data under computational constraints. Repetitive sequences have low epiplexity because they are learnable but compressible, and noise has low epiplexity because the model cannot learn from it. In contrast, data

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Epiplexity estimate (b) Data selection (c) Data generation

Figure 1: **Epiplexity for data selection and generation.** (a) Epiplexity, $S(t)$, can be estimated as the area between the learner's training loss curve and its final loss as a function of training tokens. Data which is trivial to learn or unlearnable will yield low estimates of epiplexity and result in lower performance in OOD downstream tasks. (b) For data selection, we use scaling laws to forecast the per-domain loss curves and predict the change in epiplexity, $\partial\hat{S}(t)/\partial n_j$, as a function of in-domain tokens. We then greedily maximize $\partial\hat{S}(t)/\partial n_j$ to determine the sampling weights π_k used in adaptive reweighting of the domains during data selection. (c) For data generation, we estimate $d\hat{S}(t)/dt$ as the learner's loss reduction over an evaluation buffer $\mathcal{B}$, which stores the past training data. We set $d\hat{S}(t)/dt$ as the reward and update the generator via REINFORCE to steer the generator toward producing epiplexity-maximizing synthetic data.

which makes a model work harder, through sustained and substantial decreases in loss, has high epiplexity. Further, as opposed to classical information measures, Finzi et al. [20] establishes that epiplexity can be created by computation, depends on the ordering of data, and can exceed the information content of the data-generating process itself, making it a compelling signal for interrogating the relationship between the structural information of training data and model generalization. While Finzi et al. [20] contends that data with higher epiplexity should lead to better OOD generalization, this hypothesis is largely untested. They show that existing data selection methods such as ADO [4] can *inadvertently* select for higher epiplexity data (relative to random batches), but neither develops a methodology that directly uses epiplexity for selection nor articulates the correspondence between epiplexity and downstream generalization. Relatedly, Liu et al. [21] uses epiplexity to diagnose LLM self-play and argues that sustained self-improvement requires generation pipelines that consistently increase epiplexity. However, they measure epiplexity post hoc and do not propose a way to optimize it. Thus, open questions remain on how to turn existing estimates for epiplexity into an online training objective.

In this paper, **we introduce the first approaches for optimizing epiplexity as an online training signal for both data selection and generation**. In both cases, providing an optimizable online estimator for epiplexity is nontrivial. For selection, we fit scaling laws to per-domain loss curves in order to predict the expected epiplexity gain as a function of in-domain training tokens, and we maximize this signal to choose optimal batches of training data. For generation, we first train a generator to produce epiplexity-maximizing data for a learner by defining the reward as the increase in the epiplexity of the learner's training data and apply REINFORCE policy gradients [22, 23] to optimize the generator parameters without external verifiers. Intuitively, as both trivial and noisy data have near-zero epiplexity, this objective creates data that challenges the learner while remaining predictable. We show that our proposed data selection (`EpiSelect`) and generation (`EpiGen`) algorithms both advance performance over the state of the art. Beyond our methodology, we discover a key limitation of a popular benchmark, *The Pile* [24], in evaluating data selection methods for downstream generalization. Training on its largest domain alone outperforms state-of-the-art selection methods due to its domain size imbalances. We therefore propose shifting data selection evaluation to the more balanced Common Pile dataset [25]. Finally, we also test the hypothesis that higher epiplexity corresponds to better OOD generalization and find that it holds across several settings.

2 Background

Our work integrates a diversity of research areas, including information theory and epiplexity, data selection, scaling laws, self-play, and measures for tracking learning progress.

Epiplexity Epiplexity [20] is a measure of the structural information in data that can be extracted by a model. Figure 1(a) builds intuition in showing that this quantity is small when the data is trivial to learn (rapid convergence to low loss), as in the case of simple repetitive sequences, and when the data is unlearnable (no loss decrease), as in the case of random noise. In both conditions, the data contains little to no information to help make future predictions. Conversely, this quantity is large when the data contains substantial structural information which can be gradually internalized by a model (sustained loss decrease). While natural images or highly informative text may be relatively incompressible, they contain rich structural information that enables strong downstream predictions.

The formal definition of epiplexity is the size of the model that minimizes the description length of data under a fixed compute budget. Computational constraints are important to the definition of epiplexity, because whether or not something is learnable depends on the computational resources (for example, while pseudorandom numbers are deterministically generated, they are indistinguishable from actual random numbers if we only have polynomial-time computation). More concretely, suppose P_T is a probabilistic model that has a computational budget of T. The time-bounded minimum description length (MDL) representation of data is given by the size of the model, $|P_T|$, and the size of the data X given the model, $\mathbb{E}[\log 1/P_T(X)]$. Searching over the space of programs with a fixed computational budget $\mathcal{P}_T$, the model which minimizes this two part MDL is $P_T^\star = \arg\min_{P_T \in \mathcal{P}_T} (|P_T| + \mathbb{E}[\log 1/P_T(X)])$. *Epiplexity* is precisely the size of this program $S(T) = |P_T^\star|$.

Estimating Epiplexity Finzi et al. [20] proposes using *prequential coding* to estimate epiplexity via the training curve. Suppose a randomly initialized model P_0 is trained on a stream of i.i.d. sampled tokens X_i. The total entropy code length for the training data $X_{:M} = \{X_0, \ldots, X_{M-1}\}$ and final model weights P_M is given by $L(X_{:M}, P_M) = \sum_{i=0}^{M-1} \log 1/P_i(X_i)$. Appealing to the symmetry of information, $L(Y) \approx L(X, Y) - L(X|Y)$, we can approximate the final model size $L(P_M)$, which is precisely our target definition of epiplexity, as the difference of $L(X_{:M}, P_M)$ and $L(X_{:M}|P_M)$:

$$|P_{\text{preq}}| \approx \sum_{i=0}^{M-1} \left(\log 1/P_i(X_i) - \log 1/P_M(X_i)\right). \tag{1}$$

The prequential estimate of epiplexity is given by Equation 1, which has a simple geometric interpretation. Each term can be seen as the gap between the loss incurred on a token at the moment it was seen and the loss incurred on that same token by the final model. These cumulatively sum to the area between the training loss curve and the floor set by the best model obtainable under the compute budget. We propose constructing tractable estimators for epiplexity atop this approximation of the measure. In Sections 3 and 4, we introduce two novel estimation approaches, both of which replace the final loss with the current training loss, toward greedily maximizing epiplexity gain throughout training.

Data Selection via Scaling Laws Data selection methods rest on the hypothesis that there exist curricula, or data orderings, that lead to improved model performance over random shuffles [3–5, 7, 8, 10, 11, 26]. However, good curricula are hard to discover, relying on small model forecasting and expensive proxy training runs [3, 6], or meta-learning approaches which are computationally intractable [4, 9]. Prior work has proposed using neural scaling laws to extrapolate model loss as a function of training data [4, 27, 28]. For instance, a simple power law $\hat{L}(n; \alpha, \beta, \epsilon) = \epsilon + \beta n^{-\alpha}$ reasonably models loss behavior over training when fit online. Jiang et al. [4] found that per-domain scaling laws can predict loss decreases and effectively guide data allocation across domains during training. However, this approach deferred to future work the modeling of cross-domain effects and instead relied on heuristic-based approximations. As the scaling laws trace the model's loss trajectories and epiplexity is measured as a function of that trajectory, this framework offers a natural opportunity to both apply epiplexity to the practical case of data selection among domains and to propose a parametric form for cross-domain scaling laws.

Model Self-Play Self-play has proven to support model improvement and learning without human supervision across many applications [13, 14, 29]. In self-play, two co-evolving agents each yield

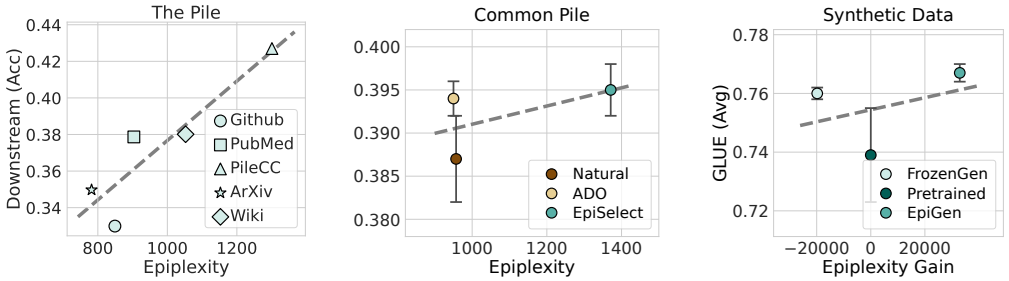


Figure 2: **Higher epiplexity is correlated with better OOD generalization across data domains, data selection methods, and synthetic data generation.** In all plots, we contrast epiplexity against downstream performance; error bars show ± 1 std over three seeds. *(Left):* We estimate the epiplexity for the largest 5 domains on the Pile by solely training models on each domain and report average accuracy on a subset of the LM Eval Harness tasks. *(Middle):* `EpiSelect` achieves higher epiplexity than alternatives and outperforms state-of-the-art curriculum learning procedures. We contrast the measured epiplexity and average zero-shot accuracy (Table 1, 124M) of `EpiSelect` on `Common Pile` against the `Natural` (no selection) baseline and `ADO` [4], a SOTA method. *(Right):* Epiplexity guided synthetic data generation improves downstream performance. We plot the average GLUE score (Table 3) of our initial checkpoint (`Pretrained`) and then compare the changes in epiplexity and performance when this checkpoint is trained on synthetic data naively generated from its original state (`FrozenGen`) versus training on synthetic data from a generator trained to maximize epiplexity (`EpiGen`).

an automatic learning curriculum which tracks the opponent's current skill level, leading to mutual improvement. Recent works have applied self-play to language model alignment [15, 16] and capability improvement [17–19]. In particular, Liu et al. [17] demonstrated that self-play on zero-sum games can effect a 10% improvement across reasoning benchmarks without the introduction of human-curated training data. The through-line goal of these works is for one agent to produce training data which is maximally informative—learnable, non-trivial, and that contains verifiable rewards—to the other agent at the current step in training. Epiplexity precisely targets this objective and offers a principled formalization of what self-play curricula aim to achieve.

Tracking Learning Progress Our implementation of epiplexity-guided synthetic data generation requires tracking the *learning progress* of a model throughout training. Rewarding a generator by learning-progress metrics is a long standing approach, dating back to literature in artificial curiosity [30–32] and active learning [33, 34]. These foundations have led to the broader paradigm of automatic curriculum learning, in which teacher models propose tasks at the limit of a learner model's competency [17, 19, 35, 36]. A failure mode in these approaches is that a generator rewarded purely for challenging the learner could collapse onto adversarial or low-entropy outputs that the learner memorizes without gaining generalizable structure [30]. To mitigate this issue, prior work proposes verifiable environments (code executor, physical constraints in robotics, or surrogate difficulty tasks) to ground the rewards in external structure [17, 19, 32, 36]. In our approach of estimating the prequential signal of epiplexity over a generation history buffer, we effectively translate the information-theoretic quantity of epiplexity into a stable REINFORCE objective while obviating the need for external verifiers.

3 EpiSelect: Maximizing Epiplexity to Select Data Domains

We introduce `EpiSelect`, a method of adaptively selecting batches of data which maximize the epiplexity through training.

3.1 Higher Epiplexity Predicts Better Downstream Generalization

We first motivate the design of our data selection method by demonstrating that higher epiplexity data domains correspond to improved downstream generalization. We train separate 124M LLaMA 2 family architecture [37] models for 15B tokens on a few size-dominant domains (arXiv, GitHub, PileCC, PubMed, and Wikipedia) in The Pile [24], a popular benchmark for curriculum learning

procedures. We limit this analysis to the five largest Pile domains since smaller domains lack sufficient in-domain tokens to train without multiple epochs (which violates the prequential estimator's assumption in Equation 1 that credits code-length reduction to unseen tokens). We compute a post-hoc estimate of the epiplexity of each domain using the prequential estimator (Equation 1) and evaluate each model on a suite of 7 zero-shot tasks (`ARC`, `HellaSwag`, `LogiQA`, `WinoGrande`, `LAMBADA`, `SciQ` and `PIQA`) in the Language Model Evaluation Harness [38]. As seen in Figure 2 (*Left*), epiplexity predicts average downstream performance across data domains (Pearson $r = 0.88$, two-tailed $p = 0.05$). In contrast, we found the model's weight norm, another scalar summary of the trained checkpoint, to show no such relationship ($r = 0.01$, Appendix A). This suggests that epiplexity captures a meaningful signal about the transferability of training data, rather than a generic property of the trained model, and motivates using it directly as a selection objective.

3.2 Tractably Estimating Per-Domain Epiplexity

While standard training uses random minibatches of data, data selection methods pursue *curriculum learning* to adaptively choose the domain distribution for each batch throughout training. Here, we propose a data selection procedure, `EpiSelect`, which computes a distribution $\pi \in \Delta^{K-1}$ over some K data domains in a given dataset in order to maximize the expected epiplexity gain during training. Following the prequential estimator of epiplexity (Equation 1), we estimate the total epiplexity $\hat{S}$ at training step t with Equation 2, where $n_k^{(s)}$ is the total number of domain k tokens seen at iteration s, and $L_m(s)$ is the loss of domain m at iteration s.

$$\hat{S}(t) = \sum_{m=1}^{K} \hat{S}_m(t) = \sum_{m=1}^{K} \sum_{s=1}^{t} (L_m(s) - L_m(t)) \tag{2}$$

In order to make online predictions for epiplexity gain during training, we draw inspiration from prior work [4] on single-domain scaling laws to propose a novel cross-domain scaling law in Equation 3, which predicts the loss on each domain m as a function of the current token counts seen from every domain. During training, we repeatedly fit the scaling law for each domain $\hat{L}_m$ to the observed per-domain losses L_m and per-domain tokens counts n_m collected from the training trajectory so far, using Equation 3. Here, α_m and β_m are positive constants, ϵ_m the irreducible error, and $\gamma_{m,k} > 0$ (with $\sum_{k=1}^{K} \gamma_{m,k} = 1$) the normalized influence of all the domains on domain m (fitting details in Appendix B). While our model includes $\gamma_{m,k}$ terms to capture cross-domain interactions, the single-domain scaling law can easily be recovered by setting $\gamma_{m,k} = 0$ for $k \neq m$. Substituting the scaling laws into Equation 2 and taking the derivative with respect to the number of tokens from domain j (full derivation for a single term in Appendix C), Equation 4 arrives at a proxy for how much additional structure would be gained by training on additional tokens from domain j at time t.

$$\hat{L}_m(n_1, \ldots, n_K) = \epsilon_m + \beta_m \left(\sum_{k=1}^{K} \gamma_{m,k} n_k\right)^{-\alpha_m} \tag{3}$$

$$\frac{\partial \hat{S}}{\partial n_k} = \sum_{m=1}^{K} \frac{\partial \hat{S}_m}{\partial n_k} = \sum_{m=1}^{K} \frac{n_m \alpha_m \gamma_{m,k}}{\sum_{i=1}^{K} \gamma_{m,i} n_i} (\hat{L}_m(n_1, \ldots, n_K) - \epsilon_m) \tag{4}$$

To maximize the overall epiplexity, we greedily maximize $\frac{\partial \hat{S}(t)}{\partial n_k}$ to select more tokens from domains which are predicted to contribute a greater increase in epiplexity. In Equation 5, we compute the distribution over domains, π_k, as a softmax over $\frac{\partial \hat{S}(t)}{\partial n_k}$, scaled by temperature parameter τ, which allows for interpolation between the uniform distribution $\tau \to \infty$ and greedy maximization of marginal epiplexity $\tau \to 0$. In our experiments, we fixed $\tau = 1$. π_k serves as the domain sampling weights when selecting the next batch of training data. The full `EpiSelect` algorithm is given in Appendix D.1.

$$\pi_k \propto \exp\left(\frac{1}{\tau} \frac{\partial \hat{S}(t)}{\partial n_k}\right) \tag{5}$$

Table 1: `EpiSelect` **matches or exceeds the strongest baseline at both model scales.** Zero-shot accuracy (%, higher is better) on LM Evaluation Harness tasks for 124M- and 1.3B-parameter models, reported as mean ± standard deviation over 3 training seeds. Highlighted cells indicate the top-performing method within each model family (all tied cells are highlighted). We compare against checkpoints trained on the `Natural` data distribution and using the state-of-the-art `ADO` selector [4].

	ARC	BBQ	CSQA	HSwag	LAM	LogiQA2	OBQA	PIQA	SciQ	WinoG	Average
124M											
`Natural`	32.8±0.5	40.6±2.6	18.9±0.9	28.0±0.1	30.7±0.7	22.9±1.3	15.7±0.4	60.8±0.4	85.5±0.8	50.7±1.7	**38.7**±0.5
`ADO`	34.0±0.3	40.1±2.1	20.5±0.6	28.1±0.1	30.9±0.7	23.9±0.4	16.1±0.5	62.3±0.4	86.5±0.7	51.7±1.0	**39.4**±0.2
`EpiSelect`	34.9±0.6	43.0±2.0	20.3±0.9	28.3±0.1	30.2±0.6	23.6±0.3	15.5±0.8	62.4±0.7	86.5±0.9	50.9±0.9	**39.5**±0.3
1.3B											
`Natural`	40.0±0.6	32.3±0.7	19.6±0.7	33.2±0.1	45.2±0.5	24.1±0.5	19.1±0.2	65.3±0.3	91.1±0.7	52.2±0.6	**42.2**±0.2
`ADO`	41.1±0.5	31.5±1.4	20.1±1.5	34.3±0.2	45.5±0.5	24.6±1.0	20.3±1.5	67.3±0.9	91.2±0.5	52.3±0.6	**42.8**±0.2
`EpiSelect`	42.0±0.4	33.4±0.2	20.3±0.5	34.3±0.2	45.5±0.7	24.9±1.2	19.7±0.2	67.8±0.5	91.6±0.3	53.2±1.1	**43.3**±0.1

3.3 Saturation of Data Selection Methods on Pile

We first trained 124M transformer models of the LLaMA 2 family [37] on the uncopyrighted subset of the Pile dataset [24], which contains around 210B tokens across 17 domains (excluding the original Books3, BookCorpus2, OpenSubtitles, YTSubtitles, and OWT2 domains due to copyright issues). However, due to the inherent domain size imbalance in the Pile uncopyrighted dataset, with one domain, PileCC, dominating at 41% of the dataset tokens, we found that the performance of data selection methods on Pile is saturated. The performance across selection methods were highly similar and could be predicted by how each method selected from PileCC. We then trained our model exclusively on the PileCC domain, and Figure 3 shows that this model was able to outperform `ADO` [4], the SOTA selection method. While it is valid that perhaps PileCC (a dataset of common crawl web text) is the most useful training data for model generalization, this result means that the ideal selector would only ever select from the PileCC domain, which makes Pile uninformative as a benchmark for distinguishing between selection strategies. Therefore, we chose to evaluate the selection methods on the newer Common Pile dataset, an 8TB collection of public-domain text. Common Pile spans 30 domains, and the largest domain (Stack V2) accounts for roughly 14% of the entire dataset. We built a token-balanced version of the Common Pile dataset (details in Appendix E) and report subsequent data selection results on this construction.

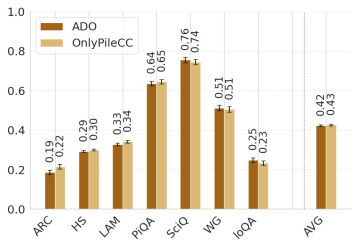


Figure 3: **The Pile does not require data selection.** Data selectors in the literature are trained to select domains on Pile, but solely training on the largest subdomain outperforms SOTA.

3.4 Data Selection Experiments

We now show that by directly selecting for domains with high epiplexity, `EpiSelect` provides a practically compelling framework for curriculum learning. In these experiments, we train decoder-only transformer models (124M and 1.3B) of the LLaMA 2 family architecture [37, 39] on the Common Pile dataset, an 8TB collection of public-domain text comprising 30 sources. We compare our method (`EpiSelect`) against a no selection baseline (`Natural`), which samples batches proportionally to domain size (balanced according to the procedure in Appendix E), and the `ADO` selection method [4], a state-of-the-art approach for curriculum learning. We train all models for a fixed budget of 15B tokens and evaluate all models on 10 common zero-shot downstream tasks in the Language Model Evaluation Harness [38]. All results are reported as mean ± standard deviation over three independent runs, each with a different random seed. Further details in Appendix F.1. Code available at `https://github.com/eysu35/EpiSelect`.

Epiplexity Maximizing Selector Achieves Highest Performance Across Zero-Shot Tasks We evaluate our epiplexity maximizing selector on a set of 10 zero-shot language understanding tasks from the LM Evaluation Harness [38]: ARC (ARC-Easy and ARC-Challenge, averaged), BBQ, CSQA, HellaSwag (HSwag), LAMBADA (LAM), LogiQA2, OBQA, PIQA, SciQ, and WinoGrande (WinoG). In Table 1, we show that for both model sizes, `EpiSelect` achieves the highest average task accuracy (39.5% and 43.3% for 124M and 1.3B, respectively), matching `ADO` at 124M and outperforming both the `Natural` (no selection) baseline and `ADO` at 1.3B by margins comparable to those reported in prior work [4]. The improvements grow with scale, with `EpiSelect` attaining

or tying for the top score on 5 of 10 tasks at 124M and 9 of 10 tasks at 1.3B. These results provide the initial support for our central hypothesis: by reweighting the batches toward domains predicted to contribute the largest marginal increase in epiplexity, the selector induces a training curriculum that produces representations which transfer more effectively to a diverse set of OOD tasks.

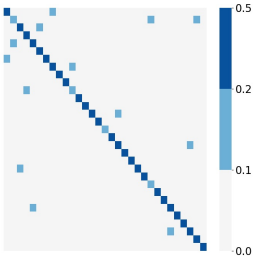


Figure 4: **Few domains interact.** Diagonal $\gamma_{m,k}$ values dominate.

Cross-Domain Effects Are Sparse and Directional

A key component of estimating epiplexity correctly is to model the cross-domain effects as seen in Equation 3. Figure 4 plots the matrix of fitted $\gamma_{m,k}$ values at the end of training, which captures the weighting that domain k tokens have when computing the predicted loss in domain m. We see that, beyond improved downstream performance, our cross-domain scaling laws surface an interpretable map of inter-domain influence. By the end of training, the fitted $\gamma_{m,k}$ matrix is diagonally dominant, which aligns with our intuition that training on domain k tokens should have the highest impact toward reducing loss on domain k. We also observe a sparse set of strong cross-domain effects, in the off-diagonal values, and note an asymmetry in the domain interactions.

4 EpiGen: Maximizing Epiplexity for Synthetic Data Generation

Next, considering cases when a training dataset is not neatly divided into domains or we require additional training data, we turn to synthetic data *generation*. For the first time, we show how epiplexity can be operationalized to select over a generator model's weights to effectively guide synthetic data generation. Intuitively, synthetic training data should challenge the learner, leading to sustained and substantial decreases in loss, while not being trivial or noise. As previewed in panel (a) of Figure 1, the prequential coding estimate of epiplexity captures this notion exactly and is well-suited to guide a generator model toward producing optimal synthetic data.

We train a generator model, P_{θ^g}, to generate high-epiplexity synthetic data used to train a separate learner model, P_θ. To estimate the epiplexity for the learner model, we implement a generation buffer $\mathcal{B}$, which accumulates the generated data throughout training, including the current batch $\mathcal{X}_t$. Our use of a buffer differs in function from experience replay [40, 41], where a buffer stores past transitions for off-policy training updates. In our method, the buffer is not used to train either model. Instead, it serves as a reference distribution for reward computation to provide a tractable estimate of epiplexity. For each batch of generated data, Equation 6 estimates the change in epiplexity as the difference between the learner loss on the samples in the buffer before ($P_{\theta_{t-1}}$) and after (P_{θ_t}) training on the current batch.

$$r_t = S(t) - S(t-1) = \sum_{x \in \mathcal{B}} \big(\log(1/P_{\theta_{t-1}}(x)) - \log(1/P_{\theta_t}(x))\big) \tag{6}$$

$$\nabla_{\theta^g} \mathcal{J} = \mathbb{E}_{\mathcal{X}_t \sim P_{\theta^g}}[(r_t - b) \cdot \sum_{x \in \mathcal{X}_t} \nabla_{\theta^g} \log P_{\theta^g}(x)] \tag{7}$$

In order for the generator to learn to maximize the resulting epiplexity, we set the reward r_t for the model as this step-wise change in epiplexity and optimize the objective $\mathcal{J}(\theta^g) = \mathbb{E}[r_t]$ using REINFORCE policy gradients [23]. In Equation 7, the gradient can be expressed as a product of an advantage term (baseline b tracks the reward history with an exponential moving average) and the log-likelihood of the batch.

4.1 Synthetic Data Generation Experiments

To empirically validate `EpiGen`, we initialize a generator and a learner model from pretrained GPT-2 weights [42]. We iteratively generate batches of synthetic data from the current generator model, train the learner for some K steps on this data, then train the generator for one step via REINFORCE policy gradients to optimize the generator for an epiplexity-maximizing distribution. The full `EpiGen` algorithm and implementation details are respectively provided in Appendices D.2 and F.2. Code available at `https://github.com/eysu35/EpiGen`.

Table 2: **Epiplexity-based reward yields highest quality synthetic data.** *Left:* perplexity of the pretrained GPT-2 checkpoint and of the learners trained on synthetic data from a frozen generator (`FrozenGen`), the epiplexity reward (`EpiGen`), and the alternative rewards (`PPL`, `NoBuffer`) on OpenWebText and two OOD corpora (lower is better). *Right:* statistics of samples drawn from each final generator, with real OpenWebText text for reference. All values are mean ± std over three seeds, where seed s sets both training and sampling. `FrozenGen`'s generator is pretrained GPT-2 in every seed, so its std is sampling variance alone. Highlighted cells mark the trained-generator method closest to the pretrained or real-text baseline.

	Learner perplexity ↓			**Generator samples**		
	OWT	**SlimPajama**	**FineWeb**	**GPT-2 ppl**	**Entropy (nats)**	**4-gram rep.**
Real text	–	–	–	25.4	–	0.034
`Pretrained` (no training)	24.7	29.8	35.1	–	–	–
`FrozenGen` (generator frozen)	26.6±0.1	31.2±0.1	37.7±0.1	63.1±0.0	4.16±0.00	0.016±0.000
Generator trained						
`EpiGen`	27.4±0.1	32.6±0.1	39.4±0.1	58.2±2.1	4.05±0.04	0.007±0.001
`PPL`	32.9±0.4	38.4±1.0	48.9±1.2	5.8±0.3	1.65±0.06	0.468±0.013
`NoBuffer`	29.0±0.6	34.4±1.2	41.8±1.2	169.7±14.5	5.07±0.11	0.002±0.001

In developing our method, we found that the epiplexity-based reward leads to REINFORCE gradients with low variance and overall stable generation dynamics (shown in Appendix G). We thus compared the trained `EpiGen` learner model against the pretrained GPT-2 checkpoint (`Pretrained`) and a set of baseline learner models trained by the following generators: a frozen generator to match the total token count (`FrozenGen`), a generator trained with a heuristic-based reward (`PPL`, negative log perplexity), and a generator trained with a learning-based reward that only considers the current batch of data (`NoBuffer`). All results are reported as mean ± standard deviation over three independent runs of the full training pipeline, each with a different random seed.

Epiplexity Reward Yields Learnable and Natural Synthetic Data Our first evaluation of `EpiGen` examines learner shift from the pretrained distribution and the quality of the generated synthetic data. For the former, we evaluate the initial learner (pretrained checkpoint) against all learners trained on synthetic data from the various generators on in- and out-of-distribution natural language datasets: OpenWebText [43] (OWT), an open source reproduction of GPT-2's pretraining dataset [42], SlimPajama [44], and FineWeb [45]. For the latter, we sample 256 512-token sequences from each final generator and measure their perplexity under pretrained GPT-2, the generator's sampling entropy, and their within-sequence 4-gram repetition rate and compare these metrics against those of 256 real OpenWebText samples. Additional evaluation details are in Appendix H.1 and Appendix H.2.

From Table 2, we draw three conclusions. First, any training on synthetic data increases perplexity on natural language. Since synthetic batches are randomly sampled from the generator at each step, this data is unsurprisingly ungrammatical, so even the `FrozenGen` learner drifts from its pretrained starting point (24.7 to 26.6 on OWT, 29.8 to 31.2 on SlimPajama, and 35.1 to 37.7 on FineWeb). Second, among the rewards that train the generator, the `EpiGen` learner best predicts natural language (OWT PPL of 27.4 vs. 32.9 and 29.0), and the same ordering holds out of distribution (SlimPajama 32.6 vs. 38.4 and 34.4; FineWeb 39.4 vs. 48.9 and 41.8), suggesting that the epiplexity-based signal guides the generator toward samples similar to real text. Third, statistics of the generated samples (GPT-2 perplexity, entropy in nats, and 4-gram repetition rate) show that the epiplexity-based reward is the only one which maintains natural-language similarity. The `PPL` generator achieves the lowest GPT-2 ppl of 5.8, but its samples are highly repetitive (4-gram rep. of 0.468 vs. `EpiGen`'s 0.007) and thus more trivial to learn. Conversely, the `NoBuffer` generator, trained directly on learner loss, produces samples with low repetition (0.002) but a high GPT-2 perplexity of 169.7 (vs. `EpiGen`'s 58.2). Here, the generator has become adversarial. To maximize its reward, it produces samples that are highly unlikely under the learner, i.e., samples that approach noise. In accordance with our hypothesis, formulating the generator's reward as an approximation of epiplexity is the only method which effectively trains the generator while keeping its samples learnable and close to natural text. For further validation, we randomly sampled from the synthetic batches during generation time and include `EpiGen` examples from steps 1000 (first sample) and 60,000 (last sample) in Appendix I.

Training on Synthetic Data Improves Downstream Generalization Next, we evaluate whether the synthetic data produced by our method improves the generalization of the learner. We follow the standard fine-tuning procedure on seven natural language understanding tasks from the GLUE benchmark [46] (CoLA, SST-2, MRPC, QQP, MNLI, QNLI, RTE), updating all weights along with a

Table 3: **GLUE transfer for all learners.** Validation scores ($\times 100$), mean $\pm$ std over three seeds. CoLA is Matthews correlation, the rest accuracy. Highlighted cells indicate the top-performing learner on each task. Every learner trained on synthetic data improves over the pretrained checkpoint by 2.0–2.8 average points, so the choice of generator should consider the learner and generator quality in Table 2.

	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Average
`Pretrained` (no training)	26.1±9.5	91.4±0.2	77.0±0.9	89.0±0.2	81.6±0.2	88.2±0.1	64.3±0.4	**73.9**±1.6
`FrozenGen` (generator frozen)	38.6±2.2	91.3±0.4	78.7±1.1	89.2±0.1	81.6±0.1	88.3±0.3	64.5±0.7	**76.0**±0.2
Generator trained								
`EpiGen`	41.5±3.2	91.6±0.4	78.3±0.4	89.1±0.1	81.5±0.2	88.4±0.2	66.4±0.7	**76.7**±0.3
`PPL`	37.5±2.6	91.1±0.4	78.5±1.0	89.1±0.2	81.7±0.2	88.3±0.3	65.0±1.3	**75.9**±0.3
`NoBuffer`	39.1±2.6	91.4±0.4	78.2±0.0	89.1±0.1	81.5±0.2	88.4±0.2	66.3±0.6	**76.3**±0.4

linear classification head, and evaluate on the development sets (details in Appendix F.2). We report each task's standard metric (Matthews correlation for CoLA, accuracy otherwise) and the average score. As shown in Table 3, the `EpiGen` learner outperforms the pretrained baseline, improving the average score by 2.8 points, with gains driven by CoLA and RTE. Critically, none of the learners in Table 3 have seen more tokens of real data than the pretrained model, so these improvements come entirely from training on synthetic data. We note that this gain is not specific to `EpiGen`. Every learner trained on additional synthetic data improved over the pretrained checkpoint to some degree. While the differences between variants are small compared to the seed variance, we read the GLUE results as preliminary evidence that training on data sampled from the pretrained generator can improve downstream generalization.

Mixtures of Synthetic and Real Data Although self-play paradigms are inherently useful in not requiring additional data to improve model performance, we conduct a data mixing experiment to assess how the epiplexity-guided synthetic data stack up against novel real data. Here, we initialize the generator model as the *trained*, epiplexity-maximizing generator from the experiments above. Then, we run the data generation pipeline with another learner model (initialized with the pretrained GPT-2 weights) and keep the generator weights frozen. In this way, we train the learner on the epiplexity-maximizing synthetic dataset for the entire duration of training, and can incrementally replace fractions of the batches of synthetic dataset with samples from the OWT dataset, all of the same sequence length, to further ablate the impact of synthetic data training on downstream performance. We make note of an additional analysis in Appendix H.3, in which we ablate the fraction of OWT data in the evaluation buffer to more realistically approximate epiplexity over all prior training data, real and synthetic. Figure 5 shows that learner models trained on 50% and 25% of synthetic data (and, subsequently, 50% and 75% of OWT data) achieve higher average performance on the OOD GLUE benchmark, indicating that training on synthetic and real data outperforms training on either one individually. This result both further validates the performance improvement we observed in Table 3 and substantiates the need for principled synthetic data generation methods.

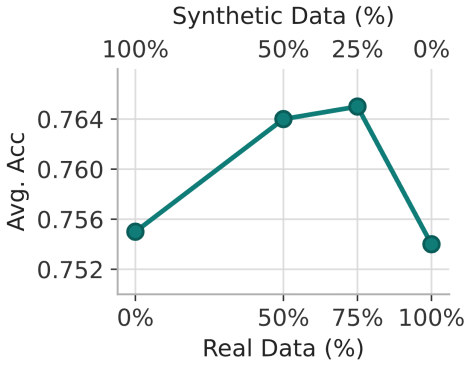


Figure 5: **Mixing synthetic and real data yields the greatest improvement**. Average GLUE score for learners trained on varying proportions of epiplexity-guided synthetic data and OWT data.

The Importance of Pretrained Initialization To better understand the efficacy of our method, we investigate the effect of model initialization to determine where the additional learning signal sources from. We ran our data generation pipeline with generator and learner model initialized with random weights and with pretrained GPT-2 weights. Figure 6 shows that, while epiplexity-guided training improves average raw GLUE performance from 0.739 to 0.767 with the pretrained GPT-2 weights, it provides essentially no change (-0.003) with the random initialization. This finding is consistent with the observation that while the ceiling of reinforcement learning is in principle not bound to the pre-trained model, empirically the quality of the pretrained initialization still constrains the performance after RL [47]. Finally, we showed in Appendix H.3 that at

matched token budgets, learners trained on epiplexity-guided synthetic data perform comparably to learners fine-tuned directly on the same number of real OpenWebText tokens (Appendix Figure 10).

5 Discussion

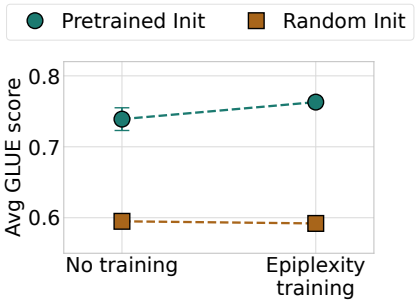


Figure 6: **Synthetic data learning signal emerges from pretrained model weights.** Average GLUE score for pretrained vs. random initialization, with and without epiplexity-guided training. Error bars show ± 1 std over three seeds for the pretrained initialization (some smaller than the markers).

Machine learning research has historically operated under the assumption that we wish to maximize generalization performance on test points drawn from the same distribution as a set of training data. From the model selection side, this setting has led to principles such as Occam's razor, where we wish to have the simplest explanation that is consistent with our observations. However, we are now entering uncharted territory, where we train models on heterogeneous data and deploy them in new unanticipated domains. This setting forces a shift in perspective from model selection to data selection. The principles here may be quite different. While we want the model for a given dataset to be as compressible as possible, perhaps we want to expose the model to data that makes the model as incompressible as possible. The rationale for this prescription is that we want our models to learn to the fullest extent, with the hope that what it is learning may be applied in a wide array of settings. This contention is at the heart of epiplexity.

We showed that epiplexity not only predicts better downstream generalization, but can be used to develop novel selection procedures that exceed state-of-the-art. Perhaps surprisingly, we also showed that epiplexity has promise as a learning signal for generating purely synthetic data that improves out-of-distribution generalization. The idea of synthesizing data for OOD generalization is entirely new, with many exciting future directions to explore. Multi-modal and multi-objective generation, as well as efficiency and scale, will be key considerations. We note some limitations of our work. First, we conducted our experiments on small models, which are inherently constrained in their performance on downstream tasks, and additional experiments should be conducted on the scalability and generalizability of these methods across model architectures. In addition, in both data selection and generation, we relied on a tractable proxy for epiplexity rather than the quantity itself, and we leave to future work the theoretical guarantees that maximizing for this proxy during training maximizes the true epiplexity. Further ahead, we will in time have a better understanding of what abstract properties of data are shared across domains beyond epiplexity, so that we can generate increasingly universal data samples, leading to broadly improved generalization.

Acknowledgements and Disclosure of Funding. We thank Eric Elmoznino, Guillaume Lajoie, Anthony GX-Chen, and all members of the Wilson Lab for helpful feedback. This work was supported by DARPA AIQ HR00112590066, NSF CAREER IIS-2145492, NSF CDS&E-MSS 2134216, Google's TPU Research Cloud (TRC) program, and Lambda's Research Grant Program. ES is supported by the Meta AIM PhD Fellowship. SQ is supported by the Two Sigma PhD Fellowship. The authors declare no competing interests.

References

[1] Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

[2] Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of llm scaling based on human-generated data. In *Forty-first International Conference on Machine Learning*, 2024.

[3] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *ArXiv*, abs/2305.10429, 2023.

[4] Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *ArXiv*, abs/2410.11820, 2024.

[5] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *ArXiv*, abs/2407.01492, 2024.

[6] Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. *ArXiv*, abs/2310.15393, 2023.

[7] Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *ArXiv*, abs/2307.14430, 2023.

[8] Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. *ArXiv*, abs/2410.07064, 2024.

[9] Yuxian Gu, Li Dong, Yaru Hao, Qingxiu Dong, Minlie Huang, and Furu Wei. Towards optimal learning of language models. *ArXiv*, abs/2402.17759, 2024.

[10] Zheng-Wen Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need. *ArXiv*, abs/2404.07965, 2024.

[11] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

[12] Jisu Kim and Juhwan Lee. Strategic data ordering: Enhancing large language model performance through curriculum learning. *ArXiv*, abs/2405.07490, 2024.

[13] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

[14] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140 – 1144, 2018.

[15] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *International Conference on Machine Learning*, 2024.

[16] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. *ArXiv*, abs/2401.10020, 2024.

[17] Bo Liu, Leon Guertler, Simon Yu, Zi-Yan Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *ArXiv*, abs/2506.24119, 2025.

[18] Pengyu Cheng, Tianhao Hu, Hang Xu, Zhisong Zhang, Yong Dai, Lei Han, and Nan Du. Self-playing adversarial language game enhances llm reasoning. *ArXiv*, abs/2404.10642, 2024.

[19] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *ArXiv*, abs/2505.03335, 2025.

[20] Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J. Zico Kolter, and Andrew Gordon Wilson. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *ArXiv*, abs/2601.03220, 2026.

[21] Wei Liu, Siya Qi, Yali Du, and Yulan He. Self-play only evolves when self-synthetic pipeline ensures learnable information gain, 2026. URL https://arxiv.org/abs/2603.02218.

[22] Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 12, pages 10887–10895, 2021.

[23] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems*, 1999.

[24] 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. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027, 2020.

[25] Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A Feder Cooper, Aviya Skowron, et al. The common pile v0. 1: An 8tb dataset of public domain and openly licensed text. *arXiv preprint arXiv:2506.05209*, 2025.

[26] Sören Mindermann, Jan Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt, 2022. URL `https://arxiv.org/abs/2206.07137`.

[27] Jared Kaplan, Sam McCandlish, Thomas Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

[28] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022.

[29] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[30] Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. *ArXiv*, abs/0812.4360, 2009.

[31] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, page 222–227, Cambridge, MA, USA, 1991. MIT Press. ISBN 0262631385.

[32] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2): 265–286, 2007. doi: 10.1109/TEVC.2006.890271.

[33] Burr Settles. Active learning literature survey, 2009.

[34] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011. URL `https://arxiv.org/abs/1112.5745`.

[35] Sainbayar Sukhbaatar, Ilya Kostrikov, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *ArXiv*, abs/1703.05407, 2017.

[36] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31:3732–3740, 2017.

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng

Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[38] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[41] Longxin Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI*, 2019.

[43] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[44] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

[45] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

[46] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

[47] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

[48] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Appendix Outline This appendix is organized as follows. In section A, we analyze the correlation between epiplexity and downstream performance, contrasting it against the model's weight norm. In section B, we provide the details of our cross-domain scaling laws, assess the accuracy of the fits, and present the ADO selector for comparison. In section C, we derive the derivative of epiplexity gain with respect to the tokens seen from a domain. In section D.1 and section D.2, we give the full `EpiSelect` and `EpiGen` algorithms. In section E, we describe the token-balancing procedure used to construct the Common Pile dataset. In section F.1 and section F.2, we provide the implementation details for the data selection and data generation experiments. In section G, we monitor the training dynamics of the epiplexity-guided generator. In section H, we analyze the learner's language modeling quality across natural language distributions and study the effect of grounding the generator reward in the evaluation buffer. In section I, we present truncated samples of the synthetic data produced by the generator.

A Epiplexity vs. Performance Analysis

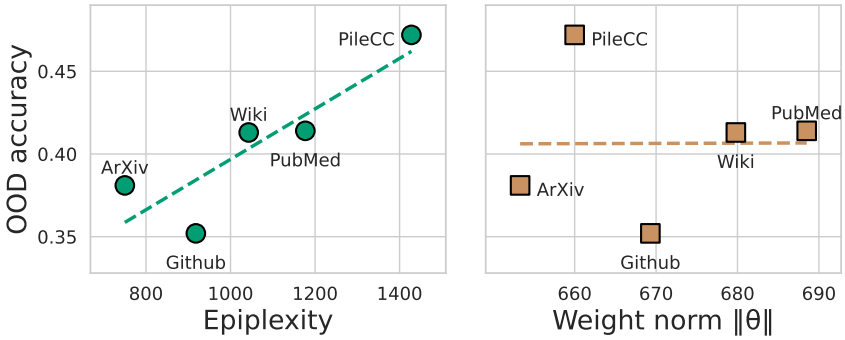


Figure 7: **Epiplexity tracks OOD downstream performance; weight norm does not.** Each point is a 124M model trained for 15B tokens on a single Pile domain, evaluated by mean zero-shot accuracy over seven held-out tasks. Epiplexity (left) is strongly correlated with accuracy (Pearson $r = 0.88$, $p = 0.05$; Spearman $\rho = 0.90$), whereas weight norm $\|\theta\|$ (right) is not ($r = 0.01$, $\rho = 0.20$). Dashed lines are least-squares fits, $n = 5$.

To check that the association between epiplexity and downstream accuracy in Section 3 reflects something specific to epiplexity, rather than any incidental property of a trained checkpoint, we compare it against the model's weight norm $\|\theta\|$. Both quantities are scalar summaries of a model trained to convergence on a single Pile domain, so if downstream accuracy were merely a generic function of "how much" the model changed during training, we would expect the weight norm to be similarly predictive.

Figure 7 shows this is not the case. Across the five domains, epiplexity is strongly associated with OOD accuracy (Pearson $r = 0.88$, Spearman $\rho = 0.90$). PileCC is highest on both, and ArXiv and GitHub are lowest for both rankings. The weight norm, by contrast, shows no relationship with downstream accuracy ($r = 0.01$, $\rho = 0.20$). With only five domains this is suggestive rather than conclusive, but it indicates that the association is not explained by the magnitude of weight change alone, and is consistent with epiplexity capturing something about the data that generic checkpoint statistics do not.

B Scaling Laws

Given some observed tokens $n_k^{(s)}$ and loss values $L_k^{(s)}$ for domains k at different points in time $s = 1, \ldots, t$, we fit a cross domain scaling law by solving the following optimization problem

$$\min_{\alpha,\beta,\epsilon,\gamma} \sum_{k,s} \text{HuberLoss}_\delta((L_k^{(s)})_{s=1}^t, (\hat{L}_k^{(s)}(\alpha,\beta,\epsilon,\gamma))_{s=1}^t)$$

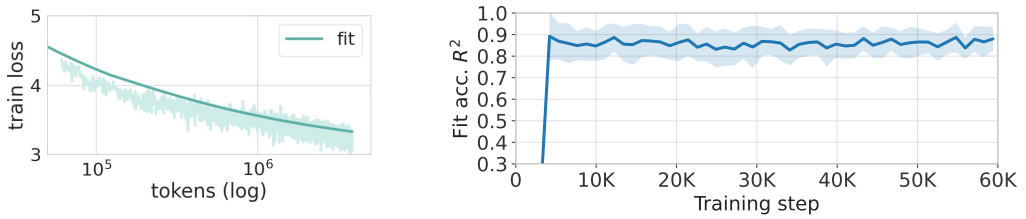


Figure 8: Panel (a): Example online cross-domain scaling-law fit for the `Common-Crawl` domain. Panel (b): Median fit accuracy over training over Common Pile domains, reported as R^2 and log-RMSE against the observed per-domain loss trajectories.

where $\hat{L}_k^{(s)} = \hat{L}_k(n_1^{(s)}, \cdots, n_K^{(s)}) = \epsilon_k + \beta_k \left(\sum_{j=1}^K \gamma_{k,j} n_j^{(s)}\right)^{-\alpha_k}$. We refit the scaling law every 1000 iterations to make use of new available observed tokens and loss values. Additionally, we optimize our parameters in the log-space, and work with α, $\log \beta$, $\log \epsilon$ and $\log \gamma$. To satisfy the simplex constraints on $\gamma_{m,k}$, which are $\gamma_{m,k} \geq 0$ and $\sum_{k=1}^K \gamma_{m,k} = 1$, we apply a softmax transformation.

We initialize all parameters with a grid derived by the Cartesian product of the following ranges per parameter: for $\alpha \in \{0.0, 0.1, \cdots, 0.8\}$, $\log \beta \in \{-2.0, -1.0, \cdots, 6.0\}$, $\log \epsilon \in \{-2.0, -1.5, \cdots, 2.0\}$. Finally, we sample $\gamma_{m,k} \sim \text{Dir}(p)$ via a Dirichlet distribution, where $p_m = 10$ and $p_k = 1$ if $k \neq m$. In other words, the Dirichlet distribution concentrates more mass on the domain which is indicated by the row index. This creates an initialization grid of 729 points which adds roughly 12 sec of overhead per fit, as opposed to the 3 seconds `ADO` takes since it does not have to fit $\gamma_{m,k}$. In practice, this increase is negligible as this overhead still constitutes less than 5% of the training time for our 124M model and less than 1% for the $1.3B$ model. Panel (a) of Figure 8 shows an example of our fitted scaling curve.

B.1 Accuracy of the Scaling-Law Fits

We assess how well the fitted scaling law reproduces the observed per-domain loss trajectories over the course of training for one of our runs. At every refit we compare the fitted function against the measured per-domain losses and report the coefficient of determination (R^2) and the root-mean-square error (RMSE) in log-loss space, matching the log-Huber objective. Panel (b) of Figure 8 shows that quality improves rapidly and then plateaus. R^2 rises to 0.9 within the first 4k steps and remains stable for the remainder of training. At the final step, the median R^2 across the 30 domains is 0.88 for the cross-domain law, with a median log-RMSE of 0.02 nats, meaning that predicted losses fall within roughly 2% of the true observation values. Three domains achieved slightly lower R^2 (DM Mathematics, Enron Emails, and Ubuntu IRC), which aligns with these domains all exhibiting near-flat, noisy loss curves, so their reduced R^2 does not indicate a problem with the fitting procedure. These results confirm that the fitted scaling laws are reliable for supporting domain selection decisions.

B.2 ADO Selector for Comparison

We provide the ADO selector for reference. ADO [4] fits a power scaling law of the form $\hat{L}(n; \alpha, \beta, \epsilon) = \epsilon + \beta n^{-\alpha}$ to each domain to inform the domain distribution π. Their preference distribution takes the form

$$\pi_k \propto \mu_k \frac{\alpha_k}{n_k} (\hat{L}_k(n_k) - \epsilon_k) \lambda_k \tag{8}$$

which is notably different from our epiplexity-maximizing objective, as they do not model the effect of the different domains on one another and introduce a heuristic credit score term λ_k, whose role is to capture some interaction between the domains.

C Estimating Epiplexity Gain from Scaling Laws

Below we derive the marginal epiplexity gain with respect to the number of tokens seen from domain k. Although Equation 2 writes the epiplexity sum over training steps t, we differentiate here with respect to the cumulative token count n_k, treating S as a function of the token counts that the step index t reaches.

$$\begin{aligned}
\frac{\partial \hat{S}_m(n_k)}{\partial n_k} &\approx \hat{S}_m(n_k+1) - \hat{S}_m(n_k) \\
&= \sum_{i_k \le n_k+1} (\hat{L}_m(i_k) - \hat{L}_m(n_k+1)) - \sum_{i_k \le n_k} (\hat{L}_m(i_k) - \hat{L}_m(n_k)) \\
&= \left[\sum_{i_k \le n_k+1} \hat{L}_m(i_k) - (n_k+1)\hat{L}_m(n_k+1) \right] - \left[\sum_{i_k \le n_k} \hat{L}_m(i_k) - (n_k)\hat{L}_m(n_k) \right] \\
&= \left[-n_k \hat{L}_m(n_k+1) \right] - \left[-(n_k)\hat{L}_m(n_k) \right] \\
&= -n_k \Delta \hat{L}_m(n_k) \approx -n_k \frac{\partial \hat{L}_m}{\partial n_k}
\end{aligned}$$

To arrive at Equation 4, we sum over all m.

D Algorithms

D.1 EpiSelect

Algorithm 1 describes how we update the domain sampling weights with `EpiSelect`.

Algorithm 1: Epiplexity Guided Data Selection

Input: Initial parameters θ, scaling law L, scaling law parameters $\alpha, \beta, \gamma, \epsilon$, loss function $\mathcal{L}$, momentum coefficient $\omega \in (0,1)$, clipping parameter δ_{min}, temperature $\tau > 0$, batch size $N \in \mathbb{N}$, and update frequency $\nu \in \mathbb{N}$.

Output: Trained parameters $\theta^{(T)}$

for $t = 1, 2, \ldots, T$ **do**

  if $\mod(t, \nu) = 0$ **then**

    $\alpha, \beta, \gamma, \epsilon$ = `FitScalingLaw` $((L^{(s)})_{s=1}^{t}, (n^{(s)})_{s=1}^{t})$

    $\hat{S}(t) = \sum_{m=1}^{K} \sum_{s=1}^{t} (\hat{L}_m^{(s)} - \hat{L}_m^{(t)})$ `// Estimate epiplexity`

    $\pi_k \propto \exp(\frac{1}{\tau} \frac{\partial \hat{S}(t)}{\partial n_k})$ `// Update distribution eq. (5)`

    $\pi_k \leftarrow \mathrm{clip}(\omega \pi_k + (1-\omega)\bar{\pi}_k, \delta_{\min})$ `// Momentum`

  $\mathcal{B} \sim \pi$, with $|\mathcal{B}| = N = \sum_k b_k^{(t)}$ `// Sample batch of data`

  $\bar{\pi}_k \leftarrow \frac{t}{t+1}\bar{\pi}_k + \frac{1}{t+1}\pi_k$ `// Global mean`

  $L_k^{(t)} = \frac{1}{n_k^{(t)}} \sum_{x \in \mathcal{B}_k} \mathcal{L}(\theta^{(t)}, x)$ `// Compute per domain losses`

  $\theta^{(t+1)} = \texttt{OptimizerUpdate}(\theta^{(t)}, \nabla_\theta \mathcal{L}(\theta^{(t)}, \mathcal{B}))$

D.2 EpiGen

Algorithm 2 describes the `EpiGen` training procedure.

E Common Pile Token Balancing

For the Common Pile dataset, we streamed up to 2B tokens for each of the 6 out of 30 largest domains and the maximum token count for the 24 out of 30 smaller domains, which totaled 29.7B tokens. Our

Algorithm 2: Epiplexity Guided Data Generation

Input: Pretrained checkpoint P_{θ_0}; learning rates η_ℓ, η_g; generation temperature τ; baseline EMA decay β; batch size B, buffer subsample size M; iterations T, learner steps per iteration K

Output: Generator P_{θ^g}, learner P_θ

$\theta^g \leftarrow \theta_0, \theta \leftarrow \theta_0$ `// Init generator, learner`

$b \leftarrow 0, \mathcal{B} \leftarrow \emptyset$ `// Init baseline, buffer`

for $t = 1, 2, \ldots, T$ **do**

- $\mathcal{X}_t \leftarrow \{x_i\}_{i=1}^{B} \overset{\text{i.i.d.}}{\sim} P_{\theta^g}(\cdot \mid \tau)$ `// Generate batch`
- $\hat{\mathcal{B}}_{t-1} \sim \mathcal{B}_{t-1}, \quad |\hat{\mathcal{B}}_{t-1}| = M$ `// Subsample buffer`
- $\hat{\mathcal{B}}_t \leftarrow \hat{\mathcal{B}}_{t-1} \cup \mathcal{X}_t$
- $L_t^{pre} \leftarrow \frac{1}{M+B} \sum_{x \in \hat{\mathcal{B}}_t} \mathcal{L}(\theta_{t-1}, x)$ `// Loss before`
- $\theta_t \leftarrow \theta_{t-1}$
- **for** $k = 1, 2, \ldots, K$ **do**
  - $\theta_t \leftarrow \theta_t - \frac{\eta_\ell}{B} \sum_{x \in \mathcal{X}_t} \nabla_{\theta_t} \mathcal{L}(\theta_t, x)$ `// Learner update`
- $L_t^{post} \leftarrow \frac{1}{M+B} \sum_{x \in \hat{\mathcal{B}}_t} \mathcal{L}(\theta_t, x)$ `// Loss after`
- $r_t \leftarrow L_t^{pre} - L_t^{post}$
- $\theta^g \leftarrow \theta^g + \frac{\eta_g}{B}[(r_t - b) \cdot \sum_{x \in \mathcal{X}_t} \nabla_{\theta^g} \log P_{\theta^g}(x)]$ `// Generator update`
- $b \leftarrow \beta \cdot b + (1 - \beta) \cdot r_t$
- $\mathcal{B}_t \leftarrow \mathcal{B}_{t-1} \cup \mathcal{X}_t$

data selection training requires 15.7B tokens over the 60,000 training steps. Thus, this construction serves as the token-balanced baseline, as truly balancing the token counts per domain would have to match the single-millions of tokens in the smallest domains, which would not provide sufficient unique tokens for training. Our `Natural` baseline, which selects proportionally to the default domain sizes, reported in Table 1 was run on this construction.

F Implementation

F.1 Data Selection

Models Our experiments use a decoder-only transformer from the LLaMA 2 family in two model parameter sizes 124M and 1.3B. The 124M model has 12 layers, 12 attention heads and an embedding size of 768. The 1.3B model has 24 layers, 16 attention heads and an embedding size of 2048. Both models use SwiGLU feed-forward layers, RMSNorm, and rotary position embeddings (RoPE). The vocabulary follows the GPT-NeoX-20B tokenizer (50,304 tokens) and the context length is 1,024 tokens. Parameters are stored in float32. Forward passes run in bfloat16. Models are implemented in JAX with Equinox for module definitions and use Optax for optimization.

Training Each model is trained for 60,000 steps with a global batch size of 256 sequences (262,144 tokens per step, 15.7B tokens total). We use AdamW with a peak learning rate of 10^{-3}, β_2 = 0.95, weight decay 10^{-4}, a 500-step linear warm-up, and cosine decay to 10^{-5}. We update the data-selection weights every 1,000 steps, after an initial 1,000 steps on the natural (by domain size) distribution to accumulate sufficient per-domain loss history for scaling-law fitting.

Compute All of our training was done on a single Cloud TPU v4-32 slice (4 host VMs, each with 8 TPU v4 chips). When launched, the model is replicated across all 32 chips, and each chip processes 8 sequences per step. Wall-clock training time is approximately 8 hours per 60,000-step run for the 124M model and 29 hours for the 1.3B model.

F.2 Data Generation

Models We initialize both the generator and learner models with pretrained GPT-2 weights (117M parameters, 12 layers, 12 attention heads, 768-dimensional embeddings), loaded via HuggingFace Transformers. The frozen reference model used for computing pretrained perplexity is a copy of the

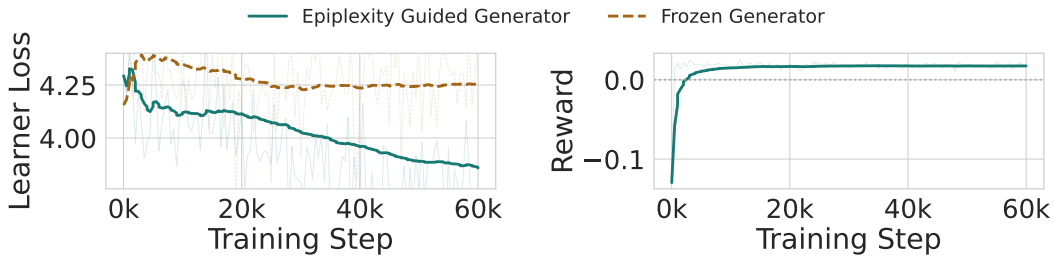


Figure 9: **Training dynamics of the epiplexity-guided generator.** (*Left*) Learner loss on generated data decreases over training for the epiplexity-guided generator, while remaining flat for the frozen generator baseline. (*Right*) REINFORCE reward increases over training, indicating that the generator learns to produce informative synthetic data throughout. No reward is computed for the frozen generator.

same model with gradients disabled. All models use the GPT-2 tokenizer (50,257 tokens). Models are implemented in PyTorch with gradient check-pointing enabled during training.

Training We use a REINFORCE-based training loop with $T = 6{,}000$ generator steps and $K = 10$ learner gradient steps per generator step, for a total of 60,000 learner updates per run. At each step the generator produces a batch of 32 sequences of 512 tokens at temperature $\tau = 1.0$. Each generated batch is appended to the accumulating buffer, which grows during training. At each step, 16 batches are sampled uniformly from the buffer and combined with the current batch to evaluate the learner. The epiplexity-based reward is computed as the decrease in learner loss before and after training on the evaluation batch. We implement gradient norm clipping (1.0) to reduce variance during the REINFORCE update. In the EMA baseline case, we use a $\beta = 0.99$. The learner uses AdamW with $\text{lr} = 10^{-5}$ and weight decay 10^{-2}, and the generator uses AdamW with $\text{lr} = 10^{-6}$ and weight decay 10^{-2}. Finally, we compute the validation loss every 1000 learner steps on 500K tokens of OpenWebText [43], which is in the original training data of GPT-2, to measure drift from the original pretraining distribution.

GLUE fine-tuning For each learner, we fine-tune all weights along with a linear classification head on the seven GLUE tasks (CoLA, SST-2, MRPC, QQP, MNLI, QNLI, RTE) for 3 epochs using a learning rate of 2×10^{-5} and batch size 32, then evaluate the model on the development set.

Compute We conduct all runs on 4 NVIDIA L40S GPUs with PyTorch DDP. Wall-clock training time is approximately 10 hours per 60,000-step run.

G Monitoring Data Generation Training Dynamics

Figure 9 shows the training dynamics (learner train loss and reward) to verify that the REINFORCE training loop functions as intended. The learner loss is measured on the current batch of generated data after K learner gradient steps. Both runs initialize from the same pretrained checkpoint and therefore start at similar losses. Over training, the epiplexity-guided generator produces data on which the learner achieves progressively lower loss, indicating that the generator learns to synthesize data more aligned with the learner's current state. The frozen generator, by contrast, produces a stationary data distribution and the learner loss remains relatively flat throughout. Meanwhile, the reward is the epiplexity signal used to update the generator via REINFORCE. The reward rises steadily over training and remains positive, confirming that the generator consistently produces data that improves the learner beyond its EMA baseline. The frozen generator is never updated, so we do not compute the reward.

H Evaluation

H.1 Held-Out Perplexity Benchmarks

Table 2 reports the perplexity of each learner on held-out natural text from three corpora. OpenWebText [43] is an open-source reproduction of GPT-2's pretraining corpus and serves as the in-distribution benchmark. SlimPajama [44] and FineWeb [45] are web-scale corpora that GPT-2 was not trained on and serve as the out-of-distribution benchmarks. Since `EpiGen` and its ablations train the learner only on synthetic text, none of the three corpora is seen during training.

Evaluation sets. For each corpus we stream documents in their released order, tokenize them with the GPT-2 BPE tokenizer, concatenate the token streams, and keep the first 5M tokens. We use the `train` split of OpenWebText, the `validation` split of SlimPajama, and the `sample-10BT` subset of FineWeb. The same fixed 5M-token slices are used for every model, so all rows of Table 2 are scored on identical tokens.

Perplexity. Each 5M-token stream is split into non-overlapping windows of 513 tokens (a 512-token context and its next-token targets), giving roughly 9.7k windows per corpus. We compute the mean next-token cross-entropy $\mathcal{L}$ in nats over all windows, in batches of 32 with no document boundaries or attention masking, and report PPL $= \exp(\mathcal{L})$. The Pretrained row scores the GPT-2 checkpoint that initializes every learner, and each remaining row scores the final learner of a training run. Learner rows are mean $\pm$ sample standard deviation over three training seeds; the Pretrained row is a single deterministic evaluation.

H.2 Generator Sample Statistics

The right-hand columns of Table 2 characterize the synthetic text that each generator produces at the end of training. For every generator we draw fresh samples exactly as during training: ancestral sampling at temperature $\tau = 1$ from the `<|endoftext|>` prompt, with no top-k or nucleus truncation, for 512 tokens per sequence. We draw 8 batches of 32 sequences (256 sequences, roughly 131k tokens) with a fixed sampling seed. The `FrozenGen` generator is never updated, so its statistics are those of pretrained GPT-2 and its standard deviation reflects sampling variance alone; all other rows are mean $\pm$ sample standard deviation over the three seeds' generators. The Real text row applies the same statistics to 256 held-out OpenWebText sequences of 512 tokens, disjoint from the evaluation sets of Appendix H.1.

GPT-2 perplexity (naturalness). The perplexity that the *pretrained* GPT-2 checkpoint assigns to the samples, $\exp$ of the mean next-token cross-entropy over all sampled tokens. Text that resembles GPT-2's training distribution scores close to the real-text value of 25.4, whereas text that is degenerate (highly repetitive) scores far below it and text that approaches noise scores far above it.

Entropy (nats). The average entropy of the generator's own next-token distribution, $\mathbb{E}_t\big[-\sum_v p_\theta(v \mid x_{<t}) \log p_\theta(v \mid x_{<t})\big]$, evaluated teacher-forced on its own samples and averaged over all positions and sequences. It measures how spread out the generator's sampling distribution is. A low value indicates a near-deterministic generator, a high value one that samples close to uniformly over the vocabulary. Entropy is a property of the generator, so it is undefined for the real-text row.

4-gram repetition. The fraction of 4-gram positions within a sequence whose 4-gram has already occurred earlier in the same sequence, pooled over all sampled sequences [48]. A value of 0 means no 4-gram is ever repeated within a sequence, and a value near 1 means the sequence loops. Real OpenWebText text has a small nonzero rate (0.034) from naturally recurring phrases.

H.3 Reward Grounding in the Evaluation Buffer

While epiplexity-guided training consistently improves downstream performance in fine-tuning based tasks, it also subjects the learner to data which diverges from natural language and leads to an increase in validation perplexity on web text. In this section, we intervene on the training algorithm by mixing fractions of OWT data into the evaluation buffer [43]. In the ideal case, we would compute the change

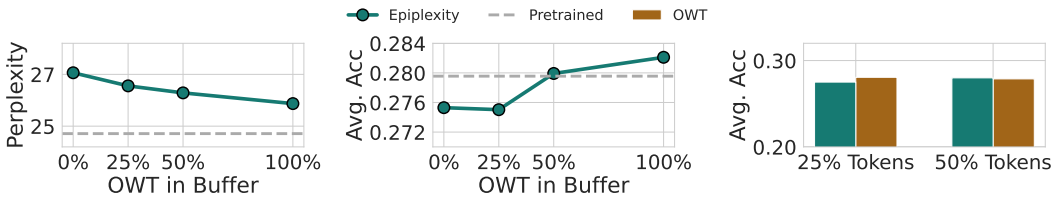


Figure 10: **Mixing OWT data into the evaluation buffer grounds the generator reward.** *(Left)* The learner's perplexity decreases monotonically as the fraction of OWT data in the buffer increases from 0% to 100%. *(Center)* Average zero-shot accuracy across 10 LM Eval Harness tasks similarly improves with more real data in the buffer. *(Right)* At matched token budgets, learners trained via epiplexity-guided synthetic data perform comparably to models fine-tuned directly on the same number of real OWT tokens.

in epiplexity as the difference in learner loss over all previous data (pretraining and synthetic) that the learner has been trained on following Equation 1. Thus, as pretraining data greatly outsizes the synthetic data, we approximate this setting by sampling from the OWT dataset when constructing our evaluation buffer to compute the reward.

Additionally, we hypothesize that the inclusion of batches of OWT data grounds the generator's predictions in its initial distribution, which reduces the effect of potential self-referential rewards. Concretely, we hold all hyperparameters fixed and ablate the proportion of OWT data in the buffer between 0%, 25%, 50%, and 100%. We evaluate each learner model on OWT perplexity as well as on the same suite of 10 zero-shot tasks from LM Evaluation Harness as in prior experiments (ARC, BBQ, BoolQ, CSQA, HellaSwag (HSwag), LAMBADA, OBQA, PIQA, SciQ, and WinoGrande) [38]. Figure 10 shows that increasing the fraction of OWT data in the buffer reduces OWT perplexity toward the pretrained GPT-2 baseline and improves average zero-shot accuracy on LM Eval Harness tasks beyond the pretrained GPT-2 baseline. Although the learner is never directly trained on real data, it is reasonable to assume being indirectly exposed to OWT data in the buffer might be driving the improvement. To measure this effect, we compared our learner to token-matched baselines where we actually train the learner on OWT data and not synthetic generations. Surprisingly, we find that the learner trained via epiplexity-maximizing synthetic data performs comparably to the corresponding baselines, which not only validates our procedure but also shows that the synthetic data from the generator is as informative as real natural web text.

I Synthetic Data Outputs

We examined the samples of synthetic data generated by the epiplexity-trained generator at step 1,000 and step 60,000 (the final step) of training and show truncated snippets of text in Figure 11. Notably, the generator receives no verifier signal and is *never* exposed to real data during either training or in the evaluation buffer in this case. The epiplexity reward is computed entirely from the learner's loss decrease over a subsample of the previous synthetic data samples. Despite this, the generator maintains consistent web-text style throughout training, producing news articles, Q&As, and blog-style content that are stylistically coherent with the OpenWebText pretraining distribution. The qualitative similarity between early and late samples suggests that the generator does not degrade or collapse under the purely self-referential reward, but instead sustains a stable generative distribution across the full training run.

Step 1,000

Maybe the bigger problem lies with this form of advertising. And perhaps I've answered everything. Save these emails and a free copy of my book instead, but just in case. Editor's note: This does not apply to tips and corrections. All parts of this article are intended for educational and informational purposes only, and do not necessarily reflect the views or opinions of TechCrunch and its parent company, Storage Spaces, Inc. Content Gaining Effectes. (a) The polar opposite. That's what happens when we assume that one form of style/coloring we use will boost effectiveness. [. . .]

The Grand Prix Jaipur is to commence this spring. What's more, the P1 class's roster includes Vandoorne, Bora Senna, Jorge Renato, Robbie Hulkenberg, Benetton Nikola Karasevic, Caleb Ewan, Angelica Hutschi, Travis Meyer, Benjamin Marko, Frank Wolff and Masaaki Matsumoto. The 2016 Series champion consists of the result of the hit Midsport Dragonslaying 20m. Bora, Senna and Jamie Stewart returned to Hamilton for Barber Motorsports' quadrupleheader series four years ago. [. . .]

Step 60,000 (Final)

Europe is "divesting itself of its remittances from Russia," Chancellor George Osborne told Parliament on Monday, the second such threat from the president of Russia since the meeting with UN delegates last December. The decision has alarmed European officials worried that Putin's economic dumping of assets to Russia is hurting thousands of EU citizens and their businesses in the troubled former Soviet bloc when Russia is perceived as something of a chess-playing player. [. . .]

Do I write a 3-D magazine or not? No. This depends on the style, strength and variety of stories. Publishing and website design are done by hand or by skilled 3-D artists that are expert in computer animation at GE. Oh, so there are of course restrictions on what people can produce in 3 direct print magazines like these "Fairyballs" and can only print a wide range of images, so "99% 2002 images" variety? True, you can print out anything from red photographs to precious candied bunches of paper alongside your old 200 page collectibles! [. . .]

Figure 11: **Truncated samples of synthetic data generated at step 1,000 (left) and step 60,000 (right).** Qualitatively, the generator produces web-text-like content throughout training (e.g. news articles, Q&As, blog posts), consistent with its OpenWebText pretraining distribution. While the content is stylistically plausible, it may be factually hallucinated.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The three claims we make in our abstract and introduction state are: (i) operationalizing epiplexity for data selection via per-domain scaling laws, (ii) operationalizing epiplexity for synthetic data generation via a REINFORCE reward, and (iii) showing that higher epiplexity corresponds to better OOD generalization. All of our claims are supported by our experiments in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Yes, section 5 explicitly discusses our limitations, including the restriction to small models (124M–1.3B parameters), and the use of a tractable proxy for epiplexity rather than the true quantity (no theoretical guarantees).

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper does not introduce new theorems, but the derivation of the marginal epiplexity gain $\partial S/\partial n_j$ used in Section 3.2 is given in full in Appendix C, with the cross-domain scaling-law assumptions stated in Section 3.2 and Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 3.2 and 4 describe the algorithms, Algorithms 1 and 2 (Appendix D.1–D.2) give pseudocode, and Appendix F provides architecture, optimizer, hyperparameter, batch size, token budget, and evaluation details for both the data-selection and data-generation experiments on publicly available datasets (CommonPile, The Pile, OpenWebText, GLUE, LM Evaluation Harness). Our code is publicly available (links in Sections 3.4 and 4.1).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is publicly available at `https://github.com/eysu35/EpiSelect` and `https://github.com/eysu35/EpiGen` (linked in Sections 3.4 and 4.1). All datasets used (The Pile, CommonPile, OpenWebText, GLUE, LM Evaluation Harness) are publicly available, and Appendix F provides sufficient implementation detail to reproduce results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not

including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix F.1 (data selection) and F.2 (data generation) specify model architecture, tokenizer, context length, optimizer, learning rates, schedule, batch size, token budgets, REINFORCE-specific hyperparameters (temperature, EMA decay, group size), and evaluation protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main results in Tables 1–3 are reported as mean $\pm$ standard deviation over three independent training runs with different random seeds, as stated in the table captions and in Sections 3.4 and 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F.1 reports that data-selection runs used a Cloud TPU v4-32 slice (32 TPU v4 chips) for approximately 8 hours per 60,000-step run, and Appendix F.2 reports that data-generation runs used 4 NVIDIA L40S GPUs with PyTorch DDP for approximately 10 hours per run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available, properly licensed text datasets (notably CommonPile, an 8TB collection of public-domain and openly licensed text), trains small-scale language models (117M, 124M), and does not involve human subjects or sensitive data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on foundational methodology for data selection and synthetic data generation in language model pretraining and does not include a dedicated broader-impacts discussion. Positive impacts include more compute-efficient training and the use of openly licensed data; potential negative impacts (e.g., that more efficient pretraining accelerates capabilities of large language models generally) are inherited from the broader language modeling literature rather than specific to this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pretrained models. The 124M-parameter models trained for evaluation are below the scale at which misuse safeguards are typically warranted, and all training data comes from existing publicly released corpora.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (The Pile, CommonPile, OpenWebText, GLUE), models (GPT-2), and software (LM Evaluation Harness) are cited via the original publications. CommonPile is explicitly described as public-domain and openly licensed text; GPT-2 and the LM Evaluation Harness are released under permissive open-source licenses. We use each within its stated terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or model checkpoints.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: While LLMs (GPT-2 and 124M-parameter LLaMA transformer) are used in our experiments, they are not required as part of our core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.