Batch-wise Convergent Pre-training: Step-by-Step Learning Inspired by Child Language Development

Ko Yoshida[1], Daiki Shiono[1], Kai Sato[1], Toko Miura [1], Momoka Furuhashi[1], Jun Suzuki[1,2,3],

[1]Tohoku University, [2]RIKEN, [3]NII LLMC,

Correspondence: yoshida.kou.p3@dc.tohoku.ac.jp

Abstract

1 Introduction

Recent language models (LMs) have achieved remarkable performance. They are typically trained on massive datasets, often containing trillions of tokens, which makes it difficult to attain comparable performance when only limited training data is available (Zhang et al., 2021; Kaplan et al., 2020). In contrast, human children are able to acquire language with far fewer words over their developmental period (Gilkerson et al., 2017), creating a substantial gap between human and machine learning efficiency (Dupoux, 2018). Bridging this gap requires developing training methods that can achieve strong language modeling performance even under limited data conditions. Such approaches are important not only for advancing natural language processing but also for providing insights into human language learning.

The BabyLM Challenge (Warstadt et al., 2023; Hu et al., 2024) was launched with this motivation in mind. Inspired by human language acquisition, it explores efficient pre-training methods under strict constraints on the amount of training data and computational resources.

Previous studies have proposed **Curriculum Learning** (Bengio et al., 2009) strategies grounded in studies of human language development. These approaches control the order in which training data is presented, starting with simpler linguistic constructs and then progressing to more complex ones. Difficulty has been defined by factors such as sentence length, punctuation count, vocabulary, syntactic complexity, and readability. Models trained under such curricula have shown improvements on some tasks compared to those trained on randomly ordered data (Hu et al., 2024).

In this study, we extend the concept of curriculum learning by combining it with a training algorithm that is designed to imitate the human step-by-step learning process, as illustrated in Figure 1. Specifically, we take inspiration from the way humans tend to master one concept before progressing to the next, while retaining previously acquired knowledge. To emulate this, our method ensures that the model continues training on a given batch until its loss falls below the pre-defined acceptable level before proceeding to the next one, thereby aiming to solidify retention and prevent forgetting of previously learned material.

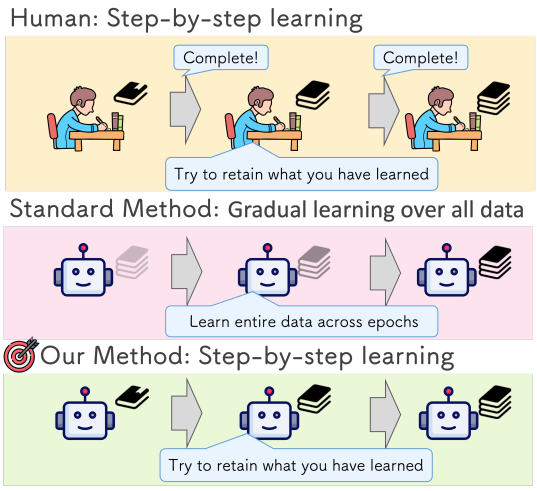

Figure 1: Inspired by human learning, our method trains language models on batches of gradually increasing difficulty while trying to retain knowledge from previous batches.

Experimental results on the BabyLM benchmark show that our method does not yet match the performance of the official baselines, particularly on grammar-oriented tasks. However, it exhibits small but consistent advantages in morphology- and discourse-level evaluations, indicating that the proposed approach affects different linguistic aspects in distinct ways under limited data conditions.

To summarize, our contributions are as follows:

- We propose a novel training algorithm that

aims to stabilize knowledge retention in LLMs by continuing training on each batch until its loss converges.

- We design a training curriculum that starts with short utterances of two to three words, extracted from child-directed speech, and progressively increases linguistic complexity, thereby mimicking stages of child language acquisition.

2 Related Work

2.1 Stages of Child Language Development

Language acquisition in children follows a gradual, stage-wise trajectory. Conti-Ramsden and Durkin (2012) provides a comprehensive overview, noting that vocabulary expands rapidly around ages one to two. By age two, children begin producing two-word combinations that gradually develop into longer, grammatically structured utterances.

In contrast, the standard training approach for large language models typically involves presenting the entire dataset at once, mixing sentences of varying complexity from the outset. This standard approach is computationally inefficient and does not reflect the incremental nature of knowledge acquisition observed in human learners.

Motivated by this discrepancy, our study proposes a curriculum that more closely mimics child language development. Instead of presenting complete sentences from the beginning, we introduce short utterances, typically two to three words as in child-directed speech incrementally, gradually increasing the linguistic complexity of the training data. This approach aims to emulate the cumulative learning process observed in early human language acquisition, where new knowledge builds upon previously acquired elements without discarding them.

2.2 Curriculum Learning Strategies for Language Models

Most curriculum learning strategies proposed in previous studies on LMs, including those in past BabyLM challenges (Warstadt et al., 2023; Hu et al., 2024) have focused on reordering existing text data based on measures of difficulty. For example, Capone et al. (2024) leveraged the observation that the first words learned by children are often highly concrete and perceptually grounded. They propose a curriculum based on lexical concreteness, categorizing the data into four stages from most to least concrete, and training the model on them sequentially.

Other approaches have used linguistic features such as sentence length, punctuation counts, or readability scores to estimate difficulty and sort the training dataset accordingly (Ghanizadeh and Dousti, 2024; Behr, 2024).

An exception is the study by Salhan et al. (2024), who explore a curriculum for masked language modeling by selectively masking different parts of speech at different stages of training. While this goes beyond simple data reordering, it still operates within the scope of conventional text and masking strategies.

Our curriculum starts with short utterances of around two to three words, as typically observed in child-directed speech, and then gradually increases linguistic complexity by expanding the utterances toward longer and more complex sentences. This approach mimics the developmental stages observed in child language acquisition, where vocabulary builds incrementally and previously acquired words are retained and reused as sentences grow in complexity.

2.3 Forgetting and Lifelong Learning

Humans and animals are capable of continuously acquiring knowledge and skills throughout their lives. However, when neural networks are trained sequentially on data drawn from different distributions, the incremental acquisition of new information generally leads to catastrophic forgetting or interference with previously acquired knowledge (French, 1999).

Research on Lifelong Learning has been extensively discussed in the context of neural networks, and a comprehensive review is provided by Parisi et al. (2019). Various approaches have been proposed to mitigate forgetting, including replay-based methods that reuse past examples, regularization-based methods that constrain parameter updates, and architectural methods that expand the network with additional neurons or layers, among others.[1]

Curriculum learning improves efficiency by ordering training data, and when combined with Lifelong Learning techniques for mitigating forgetting, it can form a framework that both accelerates learning and preserves knowledge. In this study, we integrate curriculum learning with a batch-wise

[1] See Appendix A for more details on each approach.

regularization mechanism that constrains parameter deviation while ensuring loss convergence. To our knowledge, this integration represents a novel direction in the context of pre-training small-scale language models such as BabyLM.

3 Method

Inspired by the stepwise nature of human language acquisition, we implement the learning dynamic of *advancing to the next batch only after sufficiently mastering the current one* as a constrained optimization problem. At each step t, we minimize a distance term that limits deviation from the previous parameters $\boldsymbol{W}^{(t-1)}$ while enforcing, as a *constraint*, that the cross-entropy loss on the current batch $\mathcal{X}^{(t)}$ meets a prescribed criterion. We solve this sequentially for $t = 1, 2, \ldots$, which jointly targets (i) *mastery before progression* within each batch, (ii) retention of previously acquired knowledge, and (iii) stability under small-data conditions.

We first summarize the pre-training data and curriculum (Section 3.1), and then outline the learning framework, including its *formulation* and *optimization* (Section 3.2). We describe a consolidation step based on parameter averaging that further reduces forgetting (Section 3.3). Finally, Implementation-specific choices—such as stopping rule constants, step-size clipping, hyperparameter listings, and the concrete checkpoint-averaging protocol—are summarized in Section 3.4.

3.1 Pre-training Data and Curriculum

3.1.1 Design goal.

To emulate the developmental trajectory from short utterances to full sentences and narratives, we tailor both the *data* and its *presentation schedule*. Our pre-training set integrates four sources of increasing complexity: two to three words utterances extracted from child-directed speech, slightly longer short utterances from the same source, synthetic short sentences, and finally longer child-directed narratives and dialogues. The curriculum exposes these sources in a gradually increasing order of linguistic complexity.

3.1.2 Datset components.

We compose the dataset from four sources, designed to support a smooth progression from lexical to sentential learning shown in Figure 2.

Short-utterance data. From the CHILDES dataset (Macwhinney, 2000) in the official BabyLM corpus (Choshen et al., 2024), we extract child-directed utterances of two to three words after removing metadata and non-linguistic markers. These short utterances form the starting point of our training material, corresponding to the earliest stage of child-directed input.

Category	Words	Share (%)
Short-utterance data	3,824,058	3.824
Medium-utterance data	3,399,252	3.399
Synthetic short-sentence data	12,668,607	12.669
Narrative & dialogue data	80,107,855	80.108
Total	999,999,772	100.00

Table 1: Word counts of the pre-training dataset. Each category's proportion of the total is shown in the "Share" column.

Medium-utterance data. We extract four to ten word utterances from CHILDES using the same preprocessing for short-utterance data. These examples extend minimal expressions and serve as the next stage of training material.

Synthetic short-sentence data. From English AoA ratings (Kuperman et al., 2012), we select words typically acquired by age 13 as lexical stimuli for the initial stage. For each word, Qwen3-14B (Yang et al., 2025) generates one to fifteen word sentences showing simple literal uses while excluding named entities and complex syntax. Sentences exceeding the length cap are removed, and markup and punctuation normalized, yielding concise exemplars bridging lexical and phrasal structures.

Processed narrative and dialogue data. We preprocess three child-directed sources—KidLM (Nayeem and Rafiei, 2024) (essays), TinyStories (Eldan and Li, 2023) (narratives), and TinyDialogues (Feng et al., 2024) (dialogues)—to control length and remove extraneous markup. We shorten overly long sentences (splitting or compressing while preserving meaning), remove tags and metadata, and compress redundantly verbose paragraphs. This yields streamlined narrative/dialogue material that remains semantically coherent and aligns with our staged curriculum.

3.1.3 Two-axis curriculum.

We orchestrate the presentation order along two complementary axes.

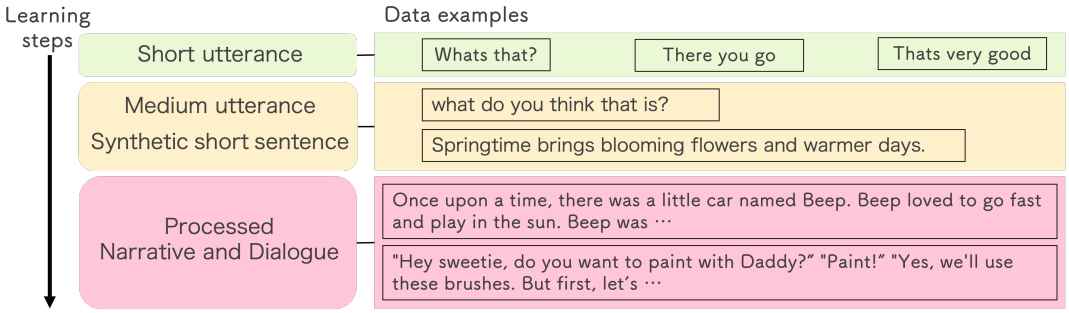

Figure 2: Overview of the staged curriculum used to construct our dataset. It progresses from short to medium CHILDES utterances, then to LLM-synthesized short sentences, and finally to processed narratives and dialogues, supporting a gradual transition from minimal expressions to full discourse.

In-batch curriculum (mixture scheduling). Within each batch, data are sampled from four sources with mixture weights that evolve during training. The share of short utterances starts high and steadily declines, medium utterances decrease more slowly, synthetic short sentences form a mid-stage peak, and narratives/dialogues increase toward the end. Let $\tau \in [0,1]$ denote normalized training progress; smooth piecewise-linear schedules $\omega_{\text{short2-3}}(\tau)$, $\omega_{\text{short4-10}}(\tau)$, $\omega_{\text{synthetic}}(\tau)$, and $\omega_{\text{long}}(\tau)$ satisfy $\sum \omega = 1$, with early-stage data decreasing, late-stage data increasing, and intermediate data forming a unimodal peak.

Through-batch curriculum (difficulty ordering). Across batches, items are sorted by estimated difficulty: CHILDES-derived short and medium utterances appear first, synthetic sentences follow by length and lexical rarity, and narratives/dialogues by a composite difficulty score(see Appendix C).

3.2 Learning Framework

We conceptualize pre-training as a sequence of batch-wise learning problems. For each batch, the model is updated repeatedly until convergence, then proceeds to the next. Each update minimizes loss while constraining large parameter deviations from the previous state. This design aims to consolidate knowledge within each batch and retain prior learning, enabling gradual, stage-wise acquisition.

3.2.1 Batch-wise Convergent Pre-training Procedure

We now present the principle and motivation of the proposed *batch-wise convergent pre-training* loop.

Let $t \in \{1, 2, \ldots, T\}$ be the time step of the pre-training process. Let $\mathcal{X}^{(t)}$ denote the t-th subset of the training data, which we usually refer to as a *batch*. Let $\phi(\boldsymbol{W}, \mathcal{X}^{(t)})$ denote the cross-entropy over the t-th batch $\mathcal{X}^{(t)}$ between the given target probability distribution $\boldsymbol{p}(x)$ and the model's distribution $\boldsymbol{q}(\boldsymbol{W}, x)$ parameterized by $\boldsymbol{W}$, where $x \in \mathcal{X}^{(t)}$:

$$\phi(\boldsymbol{W}, \mathcal{X}^{(t)}) = -\frac{1}{|\mathcal{X}^{(t)}|} \sum_{x \in \mathcal{X}^{(t)}} \boldsymbol{p}(x) \log\big(\boldsymbol{q}(\boldsymbol{W}, x)\big). \tag{1}$$

Note that $\phi(\boldsymbol{W}, \mathcal{X}^{(t)}) \geq 0$ holds.

We then iterate *inner updates* until $\phi(\boldsymbol{W}, \mathcal{X}^{(t)})$ satisfies a threshold ε. Conceptually, this corresponds to minimizing the following sequence of constrained optimization problems:

$$\boldsymbol{W}^{(t)} = \arg\min_{\boldsymbol{W}} \left\{ \frac{C}{p} \|\boldsymbol{W} - \boldsymbol{W}^{(t-1)}\|_p^p \right\} \quad \text{subject to:} \quad \phi(\boldsymbol{W}, \mathcal{X}^{(t)}) \leq \varepsilon, \tag{2}$$

where p, C and ε are hyperparameters, and $\boldsymbol{W}^{(0)}$ is the initial parameter matrix. Moreover, p represents the L_p-norm and $C \geq 0$ controls the strength of the distance term. Note that we only consider the $p \in \{1, 2\}$ cases. This formulation explicitly encodes the principle of *learning the current batch with the smallest necessary parameter change*.

Interpreting the threshold. ε corresponds to the negative log of a target average token accuracy (e.g., $-\log 0.5 = 0.6931$, $-\log 0.9 = 0.1054$), which provides a principled grounding of the constraint in probabilistic terms. This interpretation ensures that ε is not chosen arbitrarily but reflects a meaningful accuracy level; implementation-specific instantiation is deferred to Section 3.4.

Intuition and benefits. Single-pass training, with one update per batch, may progress while leaving content unlearned, causing oscillation between

forgetting and relearning. In contrast, our *progress-upon-attainment* criterion ($\phi \leq \varepsilon$) promotes stage-wise consolidation. Together with distance control described below, updates are confined to the *minimum necessary change* to reach the target.

Role of the distance term. With $p=2$, the term induces smooth *contraction* toward $\boldsymbol{W}^{t-1}$; with $p=1$, it creates per-parameter *dead bands* (i.e., effective freezing zones of width C) around $\boldsymbol{W}^{t-1}$. The former ensures stability, while the latter selectively immobilizes less important parameters.

3.2.2 Constrained Objective and Lagrangian Formulation

Building on the criterion introduced in the previous section, at each step t, our update solves a sequential constrained problem that combines *minimal deviation from the previous solution* with a *loss constraint*. Hereafter, unless otherwise specified, we focus exclusively on the inner iterative steps, which correspond to a single outer iterative step t.

Lagrangian form. Introducing a multiplier $\alpha \geq 0$ yields

$$\min_{\boldsymbol{W}} \max_{\alpha \geq 0} \mathcal{L}(\boldsymbol{W}, \alpha) = \frac{C}{p} \|\boldsymbol{W} - \boldsymbol{W}^{t-1}\|_p^p + \alpha\big(\phi(\boldsymbol{W}, \mathcal{X}^{(t)}) - \varepsilon\big). \tag{3}$$

and we alternate updates of $\boldsymbol{W}$ and α. When $\phi > \varepsilon$, α increases, strengthening the drive to reduce the data loss; after attainment, it trends back toward zero. Thus, α acts as an *on-demand gate* that intensifies learning only when needed.

3.2.3 Distance-Controlled Optimizer Design

We now detail the design of our optimizer, which incorporates distance control in a way that deliberately separates it from gradient statistics. The goal is to move from $\boldsymbol{W}^{t-1}$ to $\boldsymbol{W}$ with *only the change needed* to satisfy the current batch, thereby preserving prior knowledge.

Naive loss penalty vs. proximal correction. A straightforward approach adds the distance term directly to the data loss:

$$\underbrace{\alpha\phi(\boldsymbol{W}, \mathcal{X})}_{\text{data fit}} + \underbrace{\tfrac{C}{p} \|\boldsymbol{W} - \boldsymbol{W}^{t-1}\|_p^p}_{\text{distance control}}. \tag{4}$$

However, this contaminates the moment estimates in Adam-like optimizers. Instead, following the idea of RecAdam (Chen et al., 2020), we place distance control *outside* the loss and split the update into two stages. While RecAdam anchors parameters to the pretrained weights, we adapt this idea to our batch-wise setting by anchoring each update to the parameters from the previous batch, thereby enabling "stepwise RecAdam" updates.

1. **Data-fit update:** apply a standard optimizer, such as AdamW, to the scaled loss $\alpha\,\phi(\boldsymbol{W}, \mathcal{X})$ to obtain tentative weights $\boldsymbol{W'}$.
2. **Proximal correction:** project $\boldsymbol{W'}$ by the proximal operator of $g(\boldsymbol{W}) = \frac{C}{p}\|\boldsymbol{W} - \boldsymbol{W}^{(t-1)}\|_p^p$ to obtain the updated parameters.

This separation of roles keeps moment estimates purely data-driven, while distance control acts as a geometric post-update correction.

Scaling and gating of the data loss with α. To enforce the constraint $\phi(\boldsymbol{W}, \mathcal{X}) \leq \varepsilon$, we update the Lagrange multiplier α via projected gradient ascent. While a standard update would use a learning rate $\eta^{(i)}$, our implementation adopts a more robust approach by fixing the step size and clipping the update magnitude. This prevents numerical instability when the loss ϕ is far from the target ε. The update is given by:

$$\alpha^{(i)} = \max(0, \alpha') \tag{5}$$

$$\alpha' = \alpha^{(i-1)} + \eta^{(i)}(\phi(\boldsymbol{W}^{(i)}, \mathcal{X}) - \varepsilon) \tag{6}$$

When $\phi > \varepsilon$, α grows, amplifying the pressure to fit the data; after attainment, it relaxes toward zero. The clipping mechanism ensures that this growth is controlled. Thus, α acts as a gate that increases drive only when necessary, helping to avoid overly aggressive updates.

Two-stage update: Loss minimization and proximal projection. Let $\texttt{Optim}(\cdot)$ denote an optimizer. We can apply any optimizer; however, for example, we often choose the Adam optimizer when training LMs. We assume $\texttt{Optim}(\cdot)$ to be such an optimizer. Then, each inner iteration performs

$$\boldsymbol{W'} = \texttt{Optim}\big(\alpha^{(i)}\,\phi(\boldsymbol{W}^{(i)}, \mathcal{X})\big), \tag{7}$$

$$\begin{aligned}\boldsymbol{W}^{(i+1)} &= \operatorname{prox}_{\eta g}(\boldsymbol{W'}) \\ &= \arg\min_{\boldsymbol{W}} \Big\{ \eta\, g(\boldsymbol{W}) + \tfrac{1}{2}\|\boldsymbol{W} - \boldsymbol{W'}\|_2^2 \Big\}.\end{aligned} \tag{8}$$

Closed forms follow from g. For $p=2$,

$$\boldsymbol{W}^{(i+1)} \;=\; \frac{1}{1+C}\,\big(\boldsymbol{W}^{t-1} + \boldsymbol{W'}\big). \qquad (9)$$

i.e., a convex combination of $\boldsymbol{W'}$ and $\boldsymbol{W}^{t-1}$, with larger C pulling more strongly toward $\boldsymbol{W}^{t-1}$. For $p = 1$, the update reduces to elementwise *soft-thresholding*,

$$\boldsymbol{W}^{(i+1)} = \begin{cases} \boldsymbol{W'} - C, & \text{if } \boldsymbol{W}^{t-1} + C < \boldsymbol{W'}. \\ \boldsymbol{W}^{t-1}, & \text{if } |\boldsymbol{W'} - \boldsymbol{W}^{t-1}| \le C. \\ \boldsymbol{W'} + C, & \text{if } \boldsymbol{W'} < \boldsymbol{W}^{t-1} - C. \end{cases} \qquad (10)$$

creating a width-C *dead band* around $\boldsymbol{W}^{t-1}$ that freezes small changes.

3.2.4 Algorithmic Summary of the Training Loop

At each batch $\mathcal{X}^{(t)}$, training begins with $\alpha = 1$ and repeatedly performs inner updates. Each step computes the cross-entropy loss $\phi(\boldsymbol{W}, \mathcal{X}^{(t)})$, updates the multiplier α by a *capped* projected ascent $\alpha \leftarrow \max\{0,\ \alpha + \min(1.0,\ \eta(\phi - \varepsilon))\}$, and then applies a two-stage parameter update: (i) an AdamW step on the scaled loss $\alpha\,\phi$, and (ii) a proximal correction with respect to $\boldsymbol{W}^{(t-1)}$. The loop exits once $\alpha < 1$ and shows no increase for three consecutive steps (or when $I_{\max}$ is reached), after which $\boldsymbol{W}^{(t)}$ is committed and training advances to the next batch.

Algorithm 1: Training with Alternating Lagrangian + Proximal Update

Input: batches $\{\mathcal{X}^{(t)}\}$, init. weights $\boldsymbol{W}^{(0)}$, distance coeff. C, norm p, threshold ε , learning rate η **for** $t = 1, 2, \ldots, T$ **do**

- Initialize $\alpha \leftarrow 1$, $\boldsymbol{W} \leftarrow \boldsymbol{W}^{(t-1)}$, no_inc $\leftarrow 0$, $i \leftarrow 0$;
- **repeat**
  - (1) $\phi \leftarrow \phi(\boldsymbol{W}, \mathcal{X}^{(t)})$; $\quad i \leftarrow i + 1$;
  - (2) $\Delta \leftarrow \min\big(1.0,\ \eta(\phi - \varepsilon)\big)$; $\alpha_{\text{new}} \leftarrow \max\{0,\ \alpha + \Delta\}$; **if** $\Delta \le 0$ **then** no_inc$\leftarrow$no_inc$+1$ **else** no_inc$\leftarrow 0$;
  - (3) $\boldsymbol{W'} \leftarrow \texttt{Optim}\big(\alpha_{\text{new}} \cdot \phi\big)$;
  - (4) **if** $p = 2$ **then** $\boldsymbol{W} \leftarrow \dfrac{\boldsymbol{W'} + C\,\boldsymbol{W}^{(t-1)}}{1+C}$
  - **else if** $p = 1$ **then** $\boldsymbol{W} \leftarrow$ soft-threshold$(\boldsymbol{W'},\ \boldsymbol{W}^{(t-1)},\ C)$
  - (5) $\alpha \leftarrow \alpha_{\text{new}}$;
- **until** *($\alpha < 1$ **and** no_inc$= 3$)* **or** $i = I_{\max}$;
- Commit $\boldsymbol{W}^{(t)} \leftarrow \boldsymbol{W}$;

3.3 Parameter Averaging for Consolidation

Motivation. While our batch-wise constrained updates promote "mastery before progression," the sequence of batch endpoints $\{\overline{\boldsymbol{W}}^{(t)}\}_{t=1}^{T}$ can still exhibit oscillations as the model adapts to new material. Averaging successive solutions smooths these fluctuations, approximates an implicit ensemble, and biases the solution toward flatter minima, thereby reducing susceptibility to forgetting when exposure shifts.

Formulation. We adopt simple *arithmetic averaging* of multiple checkpoints:

$$\overline{\boldsymbol{W}} \;=\; \frac{1}{K}\sum_{j=1}^{K} \boldsymbol{W}^{(t_j)}, \qquad (11)$$

which directly yields a consolidated parameter set without introducing additional hyperparameters. This averaged model serves as a geometry-based regularizer that complements the explicit distance control, further reducing the effective drift across training.

3.4 Implementation Details

This subsection summarizes practical implementation details.

Optimizer re-initialization. Each batch is treated as an independent constrained optimization problem. Accordingly, we re-initialize the first and second moment states of the first-order optimizer at the beginning of every batch (instantiated as AdamW in our implementation). The global learning-rate schedule for model parameters is maintained consistently across training, while the Lagrange multiplier scales the loss side. This preserves a separation of roles: the optimizer explores each batch afresh, while the proximal correction preserves knowledge across batches.

Multiplier update and stopping rule. At the beginning of each batch we initialize $\alpha = 1.0$. The multiplier is updated via projected gradient ascent,

$$\alpha \leftarrow \Pi_{\alpha \ge 0}\big[\alpha + \eta\,(\phi - \varepsilon)\big],$$

with a fixed step size and clipping for numerical stability; concretely we clip the per-step increment by $\min(1.0,\ \eta(\phi - \varepsilon))$. For stopping, we monitor α rather than ϕ: convergence is declared once (i) $\alpha < 1$ and (ii) α does not increase for $M = 3$ consecutive checks, corresponding to a *stable* satisfaction of

Model	BLiMP	BLiMP-S	EWoK	GLUE	WUG-ADJ	Entity	Reading(SPR/ET)	WUG-PAST	COMPS	AoA	Avg.
GPT-BERT	80.5	73.0	52.4	70.9	41.2	39.9	3.0/8.7	27.1	59.7	22.3	43.5
GPT-2 small	74.9	63.3	51.7	54.7	50.2	31.5	3.2/7.9	7.3	56.2	5.5	36.9
Proposed(115M)	49.2	50.4	50.2	57.7	57.5	41.8	0.1/0.6	4.3	49.4	0.0	32.8
Proposed(372M)	47.8	50.5	50.5	57.7	56.2	41.2	0.0/0.5	8.5	50.6	0.3	33.1

Table 2: Main results on the BabyLM evaluation suite. We compare public baselines with our proposed models at two scales (115M and 372M model). The GPT-BERT and GPT-2 small results are directly copied from the model cards released by the BabyLM organizers. For each scale, we report the best-performing configuration selected based on the highest average score across tasks.

Model (115M)	BLiMP	BLiMP-S	EWoK	GLUE	WUG-ADJ	Entity	Reading(SPR/ET)	WUG-PAST	COMPS	AoA	Avg.
Random	64.3	57.5	50.3	57.7	26.0	18.9	0.0/3.1	15.6	53.8	0.1	31.6
Curriculum	56.6	49.6	49.6	57.7	48.1	38.1	0.6/3.7	-12.6	50.3	0.3	31.1
Curriculum-Repeat	54.1	48.2	49.8	57.7	40.6	41.3	0.8/2.4	-10.9	50.2	0.2	30.4
Proposed (p=1, C=10^{-6})	49.2	50.4	50.2	57.7	57.5	41.8	0.1/0.6	4.3	49.4	0.0	32.8
Proposed (p=2, C=1)	52.2	50.2	49.4	57.7	57.1	41.1	0.0/1.5	-2.6	50.2	0.1	32.5

Table 3: Comparison of the 115M model on BabyLM evaluation tasks between controlled baselines (Random, Curriculum, Curriculum-Repeat) and our proposed method ($p = 1, C = 10^{-6}$; $p = 2, C = 1$).

Model (372M)	BLiMP	BLiMP-S	EWoK	GLUE	WUG-ADJ	Entity	Reading(SPR/ET)	WUG-PAST	COMPS	AoA	Avg.
Random	63.5	54.1	51.5	57.7	56.7	29.9	0.3/4.1	1.4	54.1	0.1	33.9
Curriculum	54.4	48.9	50.0	57.7	70.5	28.1	0.2/3.0	-16.2	50.9	0.0	31.6
Curriculum-Repeat	52.8	48.2	49.8	57.7	49.3	41.1	0.7/2.4	6.2	50.0	0.2	32.6
Proposed (p=1, C=10^{-6})	47.8	50.5	50.5	57.7	56.2	41.2	0.0/0.5	8.5	50.6	0.3	33.1
Proposed (p=2, C=1)	52.2	47.9	50.1	57.7	60.5	40.3	0.0/1.2	2.2	50.1	0.3	33.0

Table 4: Comparison of the 372M model on BabyLM evaluation tasks between controlled baselines (Random, Curriculum, Curriculum-Repeat) and our proposed method ($p = 1, C = 10^{-6}$; $p = 2, C = 1$).

$\phi \leq \varepsilon$. As a safeguard, we cap the number of inner iterations at $I_{\max}$.

Norm and penalty coefficient. We instantiate the distance term with an L_p-norm and, in our experiments, use $p \in \{1, 2\}$ with coefficient C. For the conceptual effect of each choice (e.g., contraction vs. dead-band behavior), see Section 3.2 (*Role of the distance term*).

Hyperparameters. The main hyperparameters are the distance coefficient C, norm p, threshold ε, inner learning-rate schedule $\{\eta^{(i)}\}$ for the multiplier update, iteration cap $I_{\max}$, and the required non-increase streak M. We instantiate ε from a target token accuracy via $\varepsilon = -\log a_{\text{tgt}}$ and set 0.6931 for a_{tgt}=0.5 as an initial reference, adjusting slightly to account for label smoothing and vocabulary effects; beyond these systematic shifts, we do not tune ε empirically.

Averaging protocol. For final consolidation, we perform *arithmetic* checkpoint averaging. We save checkpoints (i) every 1M words from 1M to 9M, (ii) every 10M words from 10M to 100M, and (iii) every 100M words from 100M to 1000M, yielding 28 checkpoints in total. Their simple average defines the final weights $\overline{\boldsymbol{W}}$ used for evaluation. Throughout training, all inner-loop decisions (loss thresholding, α updates, proximal correction) operate on the live weights $\boldsymbol{W}$.

4 Experiments

This study was conducted under the *Strict* track setting of the BabyLM Challenge 2025, following the official evaluation tasks and pipeline.[2]

4.1 Evaluation Tasks

We evaluate the proposed model on the benchmarks and tasks suggested by the BabyLM challenge organizers to assess its language acquisition capabilities.footnoteSee Appendix B for more details of evaluation datasets.

[2]Details regarding the experimental setup, including tokenizer selection, model architecture, and training procedures, are provided in Appendix D.

4.2 Baselines

4.2.1 Baselines for Leaderboard Comparison

We use the official BabyLM pretrained models from both the Strict and Loose Tracks as baseline comparisons. Specifically, we set GPT-BERT (Charpentier and Samuel, 2024) and GPT-2 small (Radford et al., 2019) as baselines.

4.2.2 Baselines for Controlled Comparison

To evaluate the effectiveness of our training algorithm and curriculum design, we additionally constructed three *controlled baselines*. These models were trained with vanilla pre-training using the same dataset but under different data scheduling strategies:

Random. The model was trained for 10 epochs on the proposed dataset with sentences randomly shuffled across the entire training process.

Curriculum. The model was trained for 10 epochs on the proposed dataset while preserving the curriculum ordering of the data.

Curriculum-Repeat. The model was trained on the curriculum-ordered dataset, but with each batch repeated consecutively 10 times before moving on to the next batch.

These controlled baselines allow us to disentangle the contributions of curriculum structure and repetition effects from the impact of our proposed training algorithm.

5 Results

5.1 Main Results: Comparison with Public Baselines

Table 2 summarizes the main results on the BabyLM evaluation suite. For each model size, we report the configuration with the highest average score across all tasks. Both the 115M and 372M models use the same setting of $(p, C) = (1, 10^{-6})$. Overall, the proposed models perform lower than the official BabyLM baselines, GPT-BERT and GPT-2 small. Their average scores (around 33) remain below those baselines (43.5 and 36.9). This indicates that the constrained optimization used in our training has not yet reached the linguistic competence achieved by the official pretrained systems. The largest gaps appear in grammar-oriented benchmarks such as BLiMP and BLiMP-S. This underperformance may occur because the batch-wise constraint favors local stability over global syntactic generalization. In contrast, the models achieve relatively higher scores on WUG-ADJ and Entity Tracking, indicating stronger learning of morphological agreement and referential consistency. In summary, the proposed method provides small but consistent advantages in morphology- and discourse-level tasks, while remaining weaker on grammar-centric evaluations compared with the official baselines.

5.2 Controlled Comparison of the Proposed Method

Tables 3 and 4 compare the proposed constrained optimization with three controlled baselines that differ only in data-ordering and repetition. This analysis isolates the effect of the training algorithm itself. Overall, the proposed models achieve slightly higher average scores than the controlled baselines, but the improvements are small. For the 115M model, the proposed settings reach 32.5–32.8 in average, compared to 30–31 for the baselines. A similar pattern holds for the 372M model, where the proposed method yields around 33.0 on average, within the variance of baseline performance. These results suggest that the algorithm contributes incremental rather than substantial gains. In grammar-oriented benchmarks such as BLiMP, the proposed models tend to underperform, indicating that the constrained updates may restrict flexibility needed for broad syntactic learning. However, they show higher scores on WUG-ADJ and Entity Tracking, implying better capture of morphological and referential regularities. In summary, the proposed optimization provides minor yet consistent advantages in morphology- and discourse-level tasks, while overall performance remains comparable to the controlled baselines.

5.3 Effect of Curriculum Scheduling in the Proposed Method

Table 5 compares curriculum-ordered training with fully shuffled training data. For each model size, we report results under two representative hyperparameter settings, $(p, C) = (1, 10^{-6})$ and $(2, 1)$. Overall, the effect of curriculum scheduling is limited. Across both 115M and 372M models, the average differences between curriculum and random orders remain within one point, suggesting that the data order alone does not substantially influence final performance. In some grammar-related benchmarks such as BLiMP, the curriculum models even underperform, implying that gradual data

Model	Settings	BLiMP	BLiMP-S	EWoK	GLUE	WUG-ADJ	Entity	Reading(SPR/ET)	WUG-PAST	COMPS	AoA	Avg.
115M	p=1, Curriculum	49.2	50.4	50.2	57.7	57.5	41.8	0.1/0.6	4.3	49.4	0.0	32.8
115M	p=1, Random	53.4	48.4	49.6	57.7	62.2	41.0	0.7/1.9	-4.6	49.6	-0.4	32.7
$\|\Delta\|$		4.2↓	2.0↑	0.6↑	0.0–	4.7↓	0.8↑	0.6↓/1.3↓	8.9↑	0.2↓	0.4↑	0.1↑
115M	p=2, Curriculum	52.2	50.2	49.4	57.7	57.1	41.1	0.0/1.5	-2.6	50.2	0.1	32.5
115M	p=2, Random	51.2	50.7	50.3	57.7	57.9	41.0	0.0/1.1	5.5	50.2	0.1	33.2
$\|\Delta\|$		1.0↑	0.5↓	0.9↓	0.0–	0.8↓	0.1↑	0.0–/0.4↑	8.1↓	0.0–	0.0–	0.7↓
372M	p=1, Curriculum	47.8	50.5	50.5	57.7	56.2	41.2	0.0/0.5	8.5	50.6	0.3	33.1
372M	p=1, Random	52.8	50.6	50.0	57.7	60.7	41.6	0.1/1.2	4.2	49.7	-0.0	33.5
$\|\Delta\|$		5.0↓	0.1↓	0.5↑	0.0–	4.5↓	0.4↓	0.1↓/0.7↓	4.3↑	0.9↑	0.3↑	0.4↓
372M	p=2, Curriculum	52.2	47.9	50.1	57.7	60.5	40.3	0.0/1.2	2.2	50.1	0.3	33.0
372M	p=2, Random	54.3	49.5	49.9	57.7	59.5	40.6	0.8/3.2	0.3	50.1	-0.3	33.2
$\|\Delta\|$		2.1↓	1.6↓	0.2↑	0.0–	1.0↑	0.3↓	0.8↓/2.0↓	1.9↑	0.0–	0.6↑	0.2↓

Table 5: Ablation study of curriculum scheduling for the 115M and 372M models. We compare $p = 1$ and $p = 2$ configurations trained on curriculum-ordered versus randomly shuffled data under the same C. Each Curriculum/Random pair is followed by a $|\Delta|$ row, reporting the absolute difference with ↑/↓ indicating whether curriculum ordering increased or decreased the score, respectively.

presentation may constrain the diversity of contexts required for broader syntactic generalization. On the other hand, curriculum training yields slightly higher scores on WUG-ADJ and COMPS, indicating that staged exposure can help the model capture morphological and compositional patterns more consistently. In summary, curriculum ordering provides small and task-specific benefits but does not consistently improve overall language acquisition performance within the proposed framework.

6 Conclusion and Future Works

This study introduced a batch-wise constrained optimization framework for language model pretraining under the developmental data condition of the BabyLM Challenge 2025. Experimental results show that the method performs worse on grammar-sensitive benchmarks such as BLiMP and BLiMP-S, suggesting that the constraint limits generalization across syntactic contexts. In contrast, it yields small but consistent gains on morphology- and discourse-related tasks such as WUG-ADJ and Entity Tracking, indicating slightly improved consistency in local linguistic patterns.

Future work will address the theoretical gap between batch-wise learning and the distributional hypothesis underlying language modeling. Learning from isolated batches restricts contextual diversity, which likely explains the weak syntactic generalization observed. A promising direction is to extend the framework toward context-aggregated or memory-based architectures that integrate information across batches.

Acknowledgments

This work was supported by the "R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models" project of the Ministry of Education, Culture, Sports, Science and Technology, and JST Moonshot R&D Grant Number JPMJMS2011-35 (fundamental research). In this study, we mainly used ABCI 3.0 and the computer resource offered by Research Institute for Information Technology, Kyushu University under the category of General Projects. ABCI 3.0 is provided by AIST and AIST Solutions with support from "ABCI 3.0 Development Acceleration Use".

References

Rufus Behr. 2024. ELC-ParserBERT: Low-resource language modeling utilizing a parser network with ELC-BERT. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 140–146, Miami, FL, USA. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Arielle Borovsky, Jeffrey L Elman, and Anne Fernald. 2012. Knowing a lot for one's age: Vocabulary skill and not age is associated with anticipatory incremental sentence interpretation in children and adults. *Journal of experimental child psychology*, 112(4):417–436.

Luca Capone, Alessandro Bondielli, and Alessandro Lenci. 2024. ConcreteGPT: A baby GPT-2 based on lexical concreteness and curriculum learning. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 189–196, Miami, FL, USA. Association for Computational Linguistics.

Tyler A. Chang and Benjamin K. Bergen. 2022. Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, 10:1–16.

Lucas Georges Gabriel Charpentier and David Samuel. 2024. GPT or BERT: why not both? In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 262–283, Miami, FL, USA. Association for Computational Linguistics.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Leshem Choshen, Ryan Cotterell, Michael Y. Hu, Tal Linzen, Aaron Mueller, Candace Ross, Alex Warstadt, Ethan Wilcox, Adina Williams, and Chengxu Zhuang. 2024. [Call for Papers] The 2nd BabyLM Challenge: Sample-efficient pretraining on a developmentally plausible corpus. *arXiv preprint arXiv:2404.06214*.

Gina Conti-Ramsden and Kevin Durkin. 2012. Language development and assessment in the preschool period. *Neuropsychology Review*, 22(4):384–401.

Andrea Gregor de Varda, Marco Marelli, and Simona Amenta. 2023. Cloze probability, predictability ratings, and computational estimates for 205 english sentences, aligned with existing eeg and reading time data. *Behavior Research Methods*, 56:5190 – 5213.

Emmanuel Dupoux. 2018. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43–59.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *Preprint*, arXiv:2305.07759.

Steven Y. Feng, Noah D. Goodman, and Michael C. Frank. 2024. Is child-directed speech effective training data for language models? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 22055–22071, Miami, Florida, USA. Association for Computational Linguistics.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Mohammad Amin Ghanizadeh and Mohammad Javad Dousti. 2024. Towards data-efficient language models: A child-inspired approach to language learning. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 22–27, Miami, FL, USA. Association for Computational Linguistics.

Jill Gilkerson, Jeffrey A. Richards, Steven F. Warren, Judith K. Montgomery, Charles R. Greenwood, D. Kimbrough Oller, John H. L. Hansen, and Terrance D. Paul. 2017. Mapping the early language environment using all-day recordings and automated analysis. *American Journal of Speech-Language Pathology*, 26(2):248–265.

Akari Haga, Akiyo Fukatsu, Miyu Oba, Arianna Bisazza, and Yohei Oseki. 2024. BabyLM challenge: Exploring the effect of variation sets on language model training efficiency. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 252–261, Miami, FL, USA. Association for Computational Linguistics.

Valentin Hofmann, Leonie Weissweiler, David R. Mortensen, Hinrich Schütze, and Janet B. Pierrehumbert. 2025. Derivational morphology reveals analogical generalization in large language models. *Proceedings of the National Academy of Sciences*, 122(19):e2423232122.

Michael Y. Hu, Aaron Mueller, Candace Ross, Adina Williams, Tal Linzen, Chengxu Zhuang, Ryan Cotterell, Leshem Choshen, Alex Warstadt, and Ethan Gotlieb Wilcox. 2024. Findings of the second BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 1–21, Miami, FL, USA. Association for Computational Linguistics.

Anna A. Ivanova, Aalok Sathe, Benjamin Lipkin, Unnathi U. Kumar, Setayesh Radkani, Thomas H. Clark, Carina Kauf, Jennifer Hu, R. T. Pramod, Gabriel Grand, Vivian C. Paulun, Maria Ryskina, Ekin Akyürek, Ethan G. Wilcox, Nafisa Rashid, Leshem Choshen, Roger Levy, Evelina Fedorenko, Joshua Tenenbaum, and Jacob Andreas. 2025. Elements of world knowledge (EWoK): A cognition-inspired framework for evaluating basic world knowledge in language models. *Transactions of the Association for Computational Linguistics*, 13:1245–1270.

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

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, 44(4):978–990.

David Lopez-Paz and Marc' Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Brian Macwhinney. 2000. The childes project: tools for analyzing talk. *Child Language Teaching and Therapy*, 8.

Kanishka Misra, Julia Rayz, and Allyson Ettinger. 2023. COMPS: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2928–2949, Dubrovnik, Croatia. Association for Computational Linguistics.

Mir Tafseer Nayeem and Davood Rafiei. 2024. KidLM: Advancing language models for children – early insights and future directions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4813–4836, Miami, Florida, USA. Association for Computational Linguistics.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

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

Tracy Reuter, Arielle Borovsky, and Casey Lew-Williams. 2019. Predict and redirect: Prediction errors support children's word learning. *Developmental psychology*, 55(8):1656.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2022. Progressive neural networks. *Preprint*, arXiv:1606.04671.

Suchir Salhan, Richard Diehl Martinez, Zébulon Goriely, and Paula Buttery. 2024. Less is more: Pre-training cross-lingual small-scale language models with cognitively-plausible curriculum learning strategies. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 174–188, Miami, FL, USA. Association for Computational Linguistics.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. Superglue: Learning feature matching with graph neural networks. *Preprint*, arXiv:1911.11763.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. {LAMAL}: {LA}nguage modeling is all you need for lifelong language learning. In *International Conference on Learning Representations*.

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

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Leonie Weissweiler, Valentin Hofmann, Anjali Kantharuban, Anna Cai, Ritam Dutt, Amey Hengle, Anubha Kabra, Atharva Kulkarni, Abhishek Vijayakumar, Haofei Yu, Hinrich Schuetze, Kemal Oflazer, and David Mortensen. 2023. Counting the bugs in ChatGPT's wugs: A multilingual investigation into the morphological capabilities of a large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6508–6524, Singapore. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. 2021. When do you need billions of words of pretraining data? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1112–1125, Online. Association for Computational Linguistics.

A Overview of Prior Approaches to Catastrophic Forgetting

Replay-based methods. In natural language processing, representative replay-based methods include Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) and LAnguage MOdeling for Lifelong language learning (LAMOL) (Sun et al., 2020). Both methods aim to preserve performance on previously learned tasks when learning a new one. GEM explicitly stores examples from past tasks and constrains gradient updates so as not to reduce past task performance, whereas LAMOL avoids explicit memory by generating pseudo-samples from past tasks and mixing them with new training examples.

Regularization-based methods. Regularization is typically employed to prevent overfitting, but it has also been adapted to mitigate forgetting. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) and Synaptic Intelligence (SI) (Zenke et al., 2017) slow down the update of parameters deemed important for previously learned tasks, thereby maintaining prior knowledge while enabling new learning.

Architectural methods. Approaches such as Progressive Neural Networks (PNNs) (Rusu et al., 2022) expand the network by adding new parameters for each task. This enables the reuse of knowledge from past tasks while preventing forgetting. However, these methods require identifying which parameters correspond to each task, and the overall model size grows linearly with the number of tasks, which poses practical challenges.

Why regularization in this study? In the BabyLM setting, where the vocabulary size and available training data are restricted, replay-based methods that rely on storing or generating additional samples are less desirable. Similarly, parameter expansion approaches are impractical for complex language model architectures. For these reasons, this study adopts a regularization-based approach to mitigate forgetting.

B Details of Evaluation Tasks

BLiMP. BLiMP (Benchmark of Linguistic Minimal Pairs) (Warstadt et al., 2020) is an English zero-shot benchmark that tests the grammatical knowledge of language models. We also use BLiMP Supplement, an extended version covering dialogue, question–answer congruence, and lexical semantics.

EWoK. Ewok (Elements of World Knowledge) (Ivanova et al., 2025) assesses the extent to which LMs possess fundamental world knowledge. It uses a cognition-inspired framework to evaluate whether models can identify plausible contexts given varying fillers.

GLUE. GLUE (Wang et al., 2018) is a multi-task benchmark for evaluating natural language understanding models. As models began surpassing human performance on GLUE, the more challenging SuperGLUE (Sarlin et al., 2020) was introduced to provide a tougher evaluation.

Derivational Morphology. This task tests a language model's ability to generate English adjective-to-noun nominalizations (Hofmann et al., 2025), focusing on irregular patterns that defy simple rules.

Entity Tracking. This task measures how well LMs can track changes in the states of entities throughout a text (Kim and Schuster, 2023). Given a description of an entity's initial state and a sequence of state-altering events, the model must infer the entity's final state.

Reading time. This task assesses how well an LM's word prediction probabilities align with human cognitive processing data (de Varda et al., 2023). The task examines the relationship between model predictions and brain responses or reading behaviors.

WUG PAST. This hidden task evaluates morphological generalization by correlating model and human distributions for nonce verb past-tense and adjective nominalizations (Weissweiler et al., 2023; Hofmann et al., 2025), enabling comparison across evaluation protocols.

COMPS. This task tests whether models correctly inherit properties from superordinate to subordinate concepts using nonce-word minimal pairs (Misra et al., 2023). Models should assign higher probability to the semantically correct sentence, probing robust conceptual inheritance in language representations.

Age of Acquisition. This task estimates when models learn specific words by tracking surprisal across training (Chang and Bergen, 2022). Model-derived acquisition ages are fit with sigmoids and compared to human norms from the

Component	115M Model	372M Model
Total parameters	114,964,224	372,234,112
Number of layers	12	24
Hidden size	768	896
Attention heads	8	14
FFN intermediate size	2,688	4,864
Activation	SiLU	SiLU
Normalization	RMSNorm ($\epsilon = 10^{-6}$)	RMSNorm ($\epsilon = 10^{-6}$)
Positional encoding	RoPE ($\theta = 10^{6}$)	RoPE ($\theta = 10^{6}$)
Vocab size	16k	16k

Table 6: Architectural specifications of the 115M and 372M models. Both follow the Qwen2 decoder-only architecture with differences in scale.

MacArthur–Bates Communicative Development Inventory.

C Difficulty scoring for curriculum ordering

To rank samples, we combine five textual features:

- L_i: average sentence length (words per sentence)
- $TTR_i^{(1/2)}$: square-root adjusted type–token ratio
- R_i: rare-word rate (fraction not found in a background frequency lexicon)
- AoA_i^{mean}: mean Age of Acquisition over tokens
- $AoA_i^{\max}$: maximum Age of Acquisition over tokens

Because these features have different scales, we standardize each x_i by a z-score:

$$z(x_i) = \frac{x_i - \mu_x}{\sigma_x}. \tag{12}$$

Here, we define $\mathbf{f}_i$ and $\mathbf{w}$ as follows:

$$\mathbf{f}_i = \left(L_i,\ TTR_i^{(1/2)},\ R_i,\ AoA_i^{\text{mean}},\ AoA_i^{\max}\right),$$
$$\mathbf{w} = (0.20,\ 0.15,\ 0.20,\ 0.20,\ 0.05).$$

We define the composite difficulty score as:

$$\text{Score}_i = \sum_{k=1}^{5} w_k \cdot z(f_{k,i}), \tag{13}$$

and sort items in ascending order of Score_i. The weights prioritize structural length and lexical rarity while incorporating average and worst-case AoA to align with developmental plausibility.

D Experimental Setup

Tokenizer. In this study, we trained a new tokenizer directly on the pre-training dataset. To ensure sufficient expressiveness under the limited-data setting, we adopted the Unigram subword segmentation model. The Unigram model is known to provide flexible segmentation for infrequent and morphologically varied words, thereby balancing vocabulary compression with generalization ability (Kudo and Richardson, 2018). Concretely, we trained the tokenizer using the Hugging Face `tokenizers` implementation, with the vocabulary size fixed at 16,000. We additionally included the special tokens <pad>, <cls>, <sep>, <s>, </s>, and <unk>. For text normalization, we applied the NFKC scheme, and we enabled byte fallback to guarantee stable encoding even for inputs containing unseen characters.

Model Architecture. We adopted a Transformer-based decoder-only language model, motivated by findings that children engage in predictive sentence processing: they integrate syntactic and semantic cues to anticipate upcoming words (Borovsky et al., 2012), which in turn facilitates sentence structure learning (Reuter et al., 2019). For similar reasons, decoder-only architectures are widely used in BabyLM challenge (Haga et al., 2024; Warstadt et al., 2023; Hu et al., 2024).

Table 6 lists the architectural specifications of the two model scales used in our experiments: 115M and 372M model. Both models follow the Qwen2 (Yang et al., 2024) architecture family, sharing the same decoder-only backbone but differing in size-related hyperparameters.

Pre-training parameters. Table 7 lists the parameters that are common to both the baselines and the proposed method, covering general optimization and compute settings. Table 8 specifies the hy-

perparameters unique to our proposed method, as well as those that differ from the baselines. All experiments were trained on a single NVIDIA H200 (64GB) GPU, with a maximum wall-clock training time of approximately 24 hours per run for the proposed models.

Training hyperparameter	Setting
Total training tokens	1,176,922,330
Batch size	128
Sequence length	512
Warmup ratio	5%
Learning rate (max/min)	5e-4 / 5e-5
Scheduler	cosine
Optimizer	AdamW
AdamW $\beta_1, \beta_2, \epsilon$	0.9, 0.999, 1e-8
GPU hardware	$1 \times$ NVIDIA H200 (64GB)

Table 7: Common training hyperparameters and compute setup shared across all baselines and proposed models.

Hyperparameter	Baselines	Proposed
AdamW weight decay	0.01	0
Optimizer re-initialization	–	Enabled
p (norm type)	–	1 , 2
C (penalty coefficient)	–	10^{-6} , 1
Cross-entropy tolerance ϵ	–	0.693 $(-\ln 0.5)$
η for α update	–	0.1
max batch repeat	–	10

Table 8: Hyperparameters specific to the proposed method and those differing from the baselines.

E Additional Ablation: Optimizer Re-initialization

Table 9 presents an ablation of optimizer state re-initialization. Overall, removing re-initialization (w/o) does not consistently improve performance and sometimes leads to instability across tasks. For instance, the 115M and 372M models without re-initialization show large fluctuations in WUG-PAST and BLiMP scores, suggesting that momentum carried over between batches can introduce noise and interfere with the batch-level independence assumed by our method. At the same time, re-initialization slightly decreases scores on a few morphology-oriented benchmarks such as WUG-ADJ, indicating that the added stability may also reduce optimization flexibility. Across all settings, the differences in average scores remain within one point, confirming that the choice has only minor quantitative impact. In summary, re-initializing the optimizer state helps maintain the intended separation between batches but provides limited benefit in terms of overall downstream performance.

Model	p	Re-initialization	BLiMP	BLiMP-S	EWoK	GLUE	WUG-ADJ	Entity	Reading(SPR/ET)	WUG-PAST	COMPS	AoA	Avg.
115M	1	w/	49.2	50.4	50.2	57.7	57.5	41.8	0.1/0.6	4.3	49.4	0.0	32.8
115M	1	w/o	55.2	47.7	50.0	57.7	60.8	40.9	0.7/4.1	-13.1	50.3	0.0	32.2
$\lvert\Delta\rvert$			6.0↓	2.7↑	0.2↑	0.0–	3.3↓	0.9↑	0.6↓/3.5↓	17.4↑	0.9↓	0.0–	0.6↑
115M	2	w/	52.2	50.2	49.4	57.7	57.1	41.1	0.0/1.5	-2.6	50.2	0.1	32.5
115M	2	w/o	53.8	50.8	49.9	57.7	48.6	40.9	0.3/2.2	5.2	49.6	0.1	32.6
$\lvert\Delta\rvert$			1.6↓	0.6↓	0.5↓	0.0–	8.5↑	0.2↑	0.3↓/0.7↓	7.8↓	0.6↑	0.0–	0.1↓
372M	1	w/	47.8	50.5	50.5	57.7	56.2	41.2	0.0/0.5	8.5	50.6	0.3	33.1
372M	1	w/o	56.5	49.6	50.5	57.7	60.5	39.9	1.9/3.6	-1.0	49.8	0.1	33.5
$\lvert\Delta\rvert$			8.7↓	0.9↑	0.0–	0.0–	4.3↓	1.3↑	1.9↓/3.1↓	9.5↑	0.8↑	0.2↑	0.4↑
372M	2	w/	52.2	47.9	50.1	57.7	60.5	40.3	0.0/1.2	2.2	50.1	0.3	33.0
372M	2	w/o	52.8	46.9	50.3	57.7	57.8	41.2	0.2/2.0	-4.6	49.5	0.8	32.2
$\lvert\Delta\rvert$			0.6↓	1.0↓	0.2↑	0.0–	2.7↓	0.9↓	0.2↓/0.8↓	6.8↑	0.6↑	0.5↓	0.8↑

Table 9: Ablation study on optimizer state re-initialization for the 115M and 372M models. Each pair of rows compares training with (w/) and without (w/o) re-initializing optimizer states at the beginning of each batch. Following each pair, the $|\Delta|$ row reports the absolute difference with ↑/↓ indicating whether re-initialization increased or decreased the score, respectively.