# NEXT CONCEPT PREDICTION LEADS TO STRONGER AND MORE SCALABLE LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We propose Next Concept Prediction (NCP), a generative pretraining paradigm built on top of Next Token Prediction (NTP). NCP explicitly predicts higher-level concepts spanning multiple tokens, introducing an additional concept-level pretraining objective and using the predicted concepts to guide subsequent token generation. Our model, ConceptLM, constructs a learned concept vocabulary from its hidden states through vector quantization. A dedicated Concept Module predicts future concepts on the constrained semantic manifold defined by this vocabulary, with NCP and NTP trained jointly end-to-end. We first validate the basic framework on GPT-2 and Pythia backbones, then extend it to trillion-token pretraining on the fully open OLMo-3 architecture. With hierarchical residual routing and an adapted optimization recipe, we train an 8.9B-parameter model from scratch entirely on 5.73T tokens of open Dolma-3 data. The resulting model reaches the final pretraining loss of OLMo-3-7B using only 51.3% of its training tokens and improves the downstream average by 2.45 points, including a 5.99-point gain on GSM8K. The learned concept pathway also supports lightweight domain adaptation. Together, these results demonstrate the potential of joint token and concept prediction for scalable language-model pretraining with open data and model architectures.

## 1 INTRODUCTION

The remarkable success of modern AI is largely driven by the generative pretraining paradigm, where intelligence emerges from training on large amounts of data to predict future units (Brown et al., 2020; Ho et al., 2020). However, as models scale up, low-level learning objectives cannot fully utilize model capacity, bottlenecking AI models' potential (LeCun, 2022). The evolution toward more abstract prediction is practically evident across domains: modern computer vision methods favor predicting high-level latent representations that transcend pixel-level details (Tian et al., 2024); sequential decision-making in modern RL and robotics treats decision tasks as trajectory-level sequence modeling rather than independent, stepwise action mapping (Chen et al., 2021; Chi et al., 2024); and multimodal methods, such as LCM (team et al., 2024), elevate the predictive target to a fixed embedding space that represents whole sentences. Joint Embedding Predictive Architectures (JEPA) (Assran et al., 2023) demonstrate the efficacy of performing prediction in a latent space rather than from raw inputs and reveal the rationale for learning in a high-level space from a systematic perspective. Yet LLMs, despite reaching trillion-parameter scale, remain constrained to token-level prediction (Grattafiori & Abhimanyu Dubey, 2024; Yang et al., 2025), which may no longer be sufficient to unlock their full power.

Beyond token-level prediction, several studies have explored language modeling at other semantic levels. For instance, the Hourglass Transformer (Nawrot et al., 2022) and ContextLM (Dai et al., 2025) compress sequences to learn high-level dense representations, while H-Net (Hwang et al., 2025) and BLT (Pagnoni et al., 2024) target lower-level modeling, introduce dynamic segmentation, and are thus tokenizer-free. Despite modeling hierarchical representations, these models remain reliant on standard next-token prediction or next-byte prediction training objectives. Multi-Token Prediction (MTP) (Gloeckle et al., 2024) strengthens supervision by predicting multiple future tokens, but its prediction targets remain at the token level, and it introduces no explicit higher-level predictive signal to augment subsequent token generation. This raises a natural question: how can language models



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

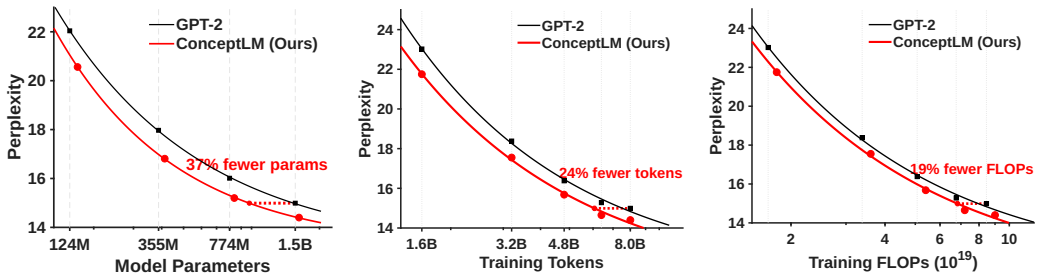



Figure 1: Scaling performance comparison of ConceptLM against traditional token-level models across three key dimensions: model parameters, training tokens, and total FLOPs. The results demonstrate that ConceptLM exhibits superior scalability.

explicitly learn to predict higher-level concepts spanning multiple tokens, and use these predictions to guide subsequent token generation?

We propose Next Concept Prediction (NCP), an additional concept-level pretraining objective, as the core of the ConceptLM framework. ConceptLM aggregates adjacent token-level hidden states into continuous concept representations and uses Vector Quantization (VQ) to construct a learned concept vocabulary. A dedicated Concept Module predicts future concepts through differentiable, normalized combinations of vocabulary vectors, constraining concept prediction to the semantic manifold defined by this vocabulary. An additional concept-level loss supervises these predictions alongside the standard next-token objective. The model then predicts subsequent tokens conditioned on the predicted concepts, rather than ground-truth future concepts, preserving causal autoregressive generation. NCP and NTP are trained jointly end-to-end.

We first validate NCP on GPT-2 and Pythia backbones trained on 8B and 300B tokens, respectively. To test whether its benefits persist at scale, we incorporate hierarchical residual routing and an adapted optimization recipe into an OLMo-based implementation, training an 8.9B-parameter model on 5.73T tokens using fully open data and a fully open backbone architecture. The resulting model reaches OLMo-3-7B's final pretraining loss with only 51.3% of its training tokens and improves the downstream average by 2.45 points.

Our main contributions are:

- **Next Concept Prediction.** We introduce NCP, an explicit pretraining objective for predicting higher-level concepts spanning multiple tokens, and develop ConceptLM to jointly learn concept prediction and token generation.
- **Validation across model scales.** Experiments on GPT-2 and Pythia backbones demonstrate improvements in language modeling and downstream performance. Parameter-matched comparisons and objective ablations examine the benefits of concept-level computation and predictive supervision.
- **Trillion-token pretraining.** We scale the framework to an 8.9B-parameter model trained on 5.73T tokens using open data and an open backbone architecture. The resulting model reaches OLMo-3-7B's final pretraining loss with only 51.3% of its training tokens and improves the downstream average by 2.45 points.

## 2 ConceptLM Architecture

In this section, we introduce the ConceptLM architecture, including the token-level encoder, the concept-level module, and the token-level decoder. ConceptLM constructs a concept-level latent space $\mathcal{D}_c$ that resides in the continuous latent space $\mathcal{H}_c$, and then performs concept-level prediction in this concept space, thereby driving the model's token prediction with conceptual supervision. Figure 2 provides an overview of the model.

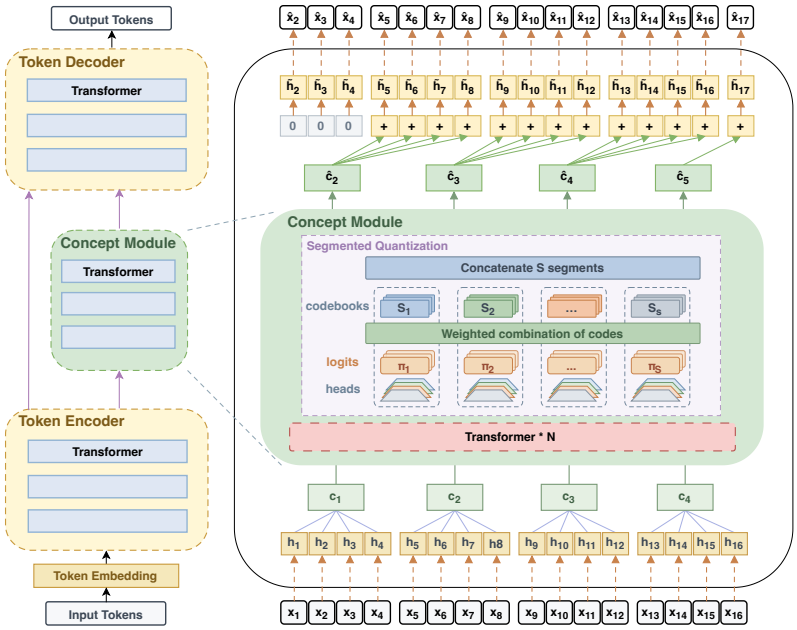



Figure 2: Overview of ConceptLM.

### 2.1 Token-level Encoder

Given a raw token sequence, the Token-level Encoder first encodes tokens into continuous token-level hidden states $h \in \mathcal{H}$, which serve as the foundational intermediate representations. These hidden states are designed for three purposes: (1) providing token-level hidden states for the Token-level Decoder to facilitate predicting the next token; (2) serving as the source for compression into concept-level hidden states for predicting the next concept; and (3) being projected into the discrete concept vocabulary space for concept-level optimization.

### 2.2 Concept-level Module

With the continuous token-level hidden states, we derive concept-level hidden states by aggregating adjacent token-level ones with a mean-pooling function $f_c : \mathcal{H} \to \mathcal{H}_c$. The length of the concept-level sequence is reduced to $T/k$, where $k$ denotes the compression factor. Then, the model predicts the next concept over a concept vocabulary defined by a Quantizer operating on the concept hidden states.

**Construct the Concept-level Vocabulary:** We employ Vector Quantization (VQ) (Gray, 1984) to transform the continuous concept-level states into a discrete and predictable space. The VQ maps continuous latent states into a finite set of representative embeddings, denoted as $f : \mathcal{H}_c \to \mathcal{D}_c$.

To enhance the representational capacity of VQ, we adopt a product quantization strategy (Jegou et al., 2010). Specifically, each continuous concept representation is partitioned into $S$ independent segments along the feature dimension. Each segment is then quantized using a separate codebook containing $N$ learnable entries $\{e_1^s, e_2^s, \ldots, e_N^s\}$ for the $s$-th segment. This results in a total discrete representation space of size $N^S$, significantly increasing the expressivity of this discrete space while maintaining a small codebook size, with a near-zero cost in terms of quantization latency.

**Predict the Next Concept:** To predict the subsequent concept, the Concept-level Module employs Transformer layers to process the historical continuous concept hidden states $\mathbf{c}_{<t/k} \in \mathcal{H}_c$. Subsequently, $S$ independent prediction heads are deployed to estimate the multi-segment discrete representation of the target concept. Specifically, each head performs a linear projection from the

hidden state dimension to the codebook size $N$, producing a logits distribution over the entries of its corresponding segment. Formally, for the $s$-th segment,

$$w_n^s = P_{\theta_c^s}(h_{<t/c}), \quad n = 1, \ldots, N,$$

where $N$ is the number of codebook entries for this segment, and $\theta_c^s$ denotes the shared Transformer layers with a segment-specific prediction head.

Rather than performing hard sampling, which would introduce non-differentiability, the predicted segment representation is derived as a weighted sum of the codebook entries based on these predicted logits, noted by:

$$\hat{c}^s = \sum_{n=1}^{N} w_n^s \, e_n^s.$$

The final predicted conceptual representation $\hat{c}$ is the concatenation of all segment-wise predictions:

$$\hat{c} = \text{concat}(\hat{c}^1, \hat{c}^2, \ldots, \hat{c}^S).$$

### 2.3 Token-level Decoder

Finally, the Decoder performs autoregressive generation to predict the next token. To align with the number of corresponding token hidden states of each concept, the predicted concept representation $\hat{\mathbf{c}}$ is broadcast $k$ times to match the token-level hidden states. This broadcast concept representation is then integrated with the token-level hidden states via element-wise addition.

**Prevent Information Leakage:** Since each concept spans multiple tokens, predicting the next concept requires access to the complete Encoder hidden states $h$ of the previous concept. This leads to a misalignment in which the first $k-1$ tokens of a sequence lack a predicted concept. To prevent information leakage, we shift the broadcast concept sequence forward by $k-1$ positions before the element-wise addition, adding zero vectors to the first $k-1$ tokens.

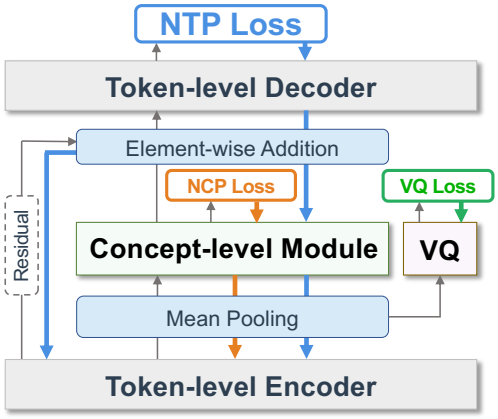


Figure 3: ConceptLM forward and backward details.

This element-wise addition can be simply described by:

$$\tilde{h}_t = \begin{cases} h_t + \mathbf{0}, & 0 \le t < k-1 \\ h_t + \hat{\mathbf{c}}_{\lfloor t/(k-1) \rfloor}, & k-1 \le t < T \end{cases}$$

The Token-level Decoder with parameter $\theta$ predicts the next token based on the fused representation, formally:

$$p(x_{t+1} \mid x_{\le t}) = P_\theta(\tilde{h}_{<t}) \, .$$

## 3 ConceptLM Training

We delineate the training objectives of ConceptLM, including $\mathcal{L}_{\text{VQ}}$, which constructs the discrete concept vocabulary; $\mathcal{L}_{\text{NCP}}$, which supervises next concept prediction; and $\mathcal{L}_{\text{NTP}}$, which supervises next token prediction. We show the forward and backward details in Figure 3.

**The Supervision Signal to Construct Concept Vocabulary:** $\mathcal{L}_{\text{VQ}}$ is strictly confined to the Quantizer module without affecting the model's representation learning. By isolating this loss, we ensure the codebook entries effectively track the latent space without imposing quantization-specific constraints back onto the hidden representations. Formally, the VQ loss is defined as:

$$\mathcal{L}_{\text{VQ}} = \|\text{sg}(c) - d\|_2^2 + \beta \|c - \text{sg}(d)\|_2^2,$$

where $c \in \mathcal{H}_c$ denotes the input concept-level hidden state, $d \in \mathcal{D}_c$ is the corresponding quantized embedding, and $\text{sg}(\cdot)$ denotes stop-gradient.

**The Supervision Signal to Predict Next Concept:** Building upon this discrete space, $\mathcal{L}_{\text{NCP}}$ supervises the NCP LM to predict the next concept. To supervise this predictive process, we employ a Mean Squared Error (MSE) loss that aligns the predicted concept $\hat{c}$ with the ground-truth continuous latent states $h_c$. It is calculated over all predicted time steps:

$$\mathcal{L}_{\text{NCP}} = \frac{1}{M-1} \sum_{t=1}^{M-1} \|\hat{c}_t - c_t\|_2^2, M = T/k.$$

By regressing to a weighted combination of codebook representations, the $\mathcal{L}_{\text{NCP}}$ effectively supervises next concept prediction while avoiding unconstrained optimization. The context prediction gradients propagate back to the Encoder, forcing the Encoder to capture high-level semantic and long-range dependencies that are essential for predicting concepts, rather than merely focusing on tokens.

**The Supervision Signal to Predict Next Token:** With the fused token-level representation $\tilde{h}$, the token-level supervision can be formulated by:

$$\begin{aligned}\mathcal{L}_{\text{NTP}} &= -\sum_{t=0}^{T-1} \log P(x_{t+1} \mid x_{\leq t}, \hat{c}_{\leq \lfloor t/(k-1) \rfloor}) \\ &= -\sum_{t=1}^{T-1} \log P(x_{t+1} \mid \tilde{h}_t)\end{aligned}$$

where the conditional dependency $(x_{\leq t}, \hat{c}_{\leq \lfloor t/(k-1) \rfloor})$ is instantiated by the integrated representation $\tilde{h}_t$. $\mathcal{L}_{\text{CE}}$ provides dense, end-to-end supervision. It affects all trainable parameters through backpropagation, ensuring the model maintains its fundamental language modeling capabilities.

**Total Training Objective of ConceptLM:** Finally, we train the model end-to-end with the total training objective:

$$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \mathcal{L}_{\text{NCP}} + \mathcal{L}_{\text{VQ}}.$$

# 4 Experiments

To evaluate ConceptLM, we conduct training-from-scratch experiments across GPT-2 and Pythia backbones at various parameter scales. Section 4.1 details the general experimental setup and the configurations for ConceptLM. Section 4.2 shows the scaling experiments for the GPT-2 (**?**) backbone, from 160M to 1.5B parameters with 8B tokens, and for the Pythia (Biderman et al., 2023) backbone, from 70M to 7B parameters with 300B tokens.

## 4.1 Experimental Setup

**Models and data.** We evaluate the basic ConceptLM framework on GPT-2 and Pythia backbones, trained from scratch on 8B OpenWebText tokens and 300B Pile tokens, respectively. The GPT-2 backbone sizes are 124M, 355M, 774M, and 1.5B. The Pythia experiments cover backbones from 70M to 7B. We then evaluate an OLMo-based extension trained from scratch on 5.73T tokens of Dolma 3 Mix. Detailed settings and the complete large-scale results appear in Appendix F.

**Model configurations.** We apply the two-layer Concept Module and $N = 64$ configuration to the GPT-2/Pythia experiments, with $k = 4$ and segment counts matching each backbone's attention heads. For VQ, to avoid representation collapse, we draw inspiration from SimVQ (Zhu et al., 2025) by introducing a two-layer MLP with ReLU activation to transform the codebook embeddings. For the large-scale OLMo model, we use 16 Encoder layers, 8 Concept Module layers, and 16 Decoder layers, with hidden size 4,096, 32 attention heads, and context length 8,192. Its concept vocabulary has $S = 32$ segments and $N = 128$ entries per segment, with $k = 4$. The resulting model has approximately 8.94B parameters. To improve overall model performance, we further refine the residual connections and optimization recipe, as detailed in Appendix F.

**Evaluation and baselines.** The GPT-2/Pythia experiments retain the language-modeling and downstream evaluations the same as in the original work and ContextLM. The OLMo-based model follows the OLMo evaluation protocol and is compared with the released OLMo-3-7B Stage-1 reference. The overall score is the unweighted mean of 26 constituent benchmark scores; bits per byte (BPB) is evaluated and averaged separately.

### 4.2 PRETRAIN CONCEPTLM FROM SCRATCH

We evaluate our method from a scaling perspective using GPT-2 and Pythia backbones. For each trained model, we employ a diverse suite of benchmarks: 9 downstream tasks (Lambada OpenAI/Standard (Paperno et al., 2016), ARC-Easy/Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2019), PIQA (Bisk et al., 2019), HellaSwag (Zellers et al., 2019), SciQ (Welbl et al., 2017), RACE (Lai et al., 2017)) and 4 language modeling tasks, covering Lambada, Wikitext (Merity et al., 2016), and the original training distributions [1].

**GPT-2 Scaling Experiments:** For the GPT-2 series, we train three configurations from scratch for comparison: (i) Baseline: The standard GPT-2; (ii) ConceptLM: Our method; (iii) Parameter-Matched (PM): The baseline with an additional two token-level layers to match the parameter count of ConceptLM. Since we use very small codebooks, which have fewer than 2M parameters in total in the largest XL model, we do not match these parameters. We evaluate these models in scaling metrics, which we visualize in Figure 1. Compared to the base models, we achieve the same performance with the largest base models using only 76% of the tokens or 63% of the parameters. And our model gains the improvement with significantly lower computational cost, requiring only 81% used by the standard models.

| Model | O/P | Wiki | L(O) | L(S) | Avg PPL ↓ |
|---|---|---|---|---|---|
| GPT2-124M | 22.04 | 43.99 | 69.29 | 272.43 | 101.94 |
| GPT2-PM | 21.22 | 42.15 | 60.05 | 273.12 | 99.14 |
| ContextLM | 20.68 | 41.45 | 55.22 | 231.41 | 87.19 |
| **ConceptLM** | **20.56** | **40.86** | **51.83** | **225.74** | **84.75**↓ 17.19 |
| GPT2-355M | 17.97 | 34.24 | 35.54 | 122.54 | 52.57 |
| GPT2-PM | 17.67 | 33.59 | 32.91 | 126.03 | 52.55 |
| ContextLM | 17.03 | 32.08 | 28.06 | 95.06 | 43.06 |
| **ConceptLM** | **16.81** | **31.53** | **26.96** | **86.75** | **40.51**↓ 12.06 |
| GPT2-774M | 16.01 | 30.03 | 24.58 | 74.76 | 36.35 |
| GPT2-PM | 15.89 | 29.71 | 23.50 | 62.45 | 32.89 |
| ContextLM | 15.41 | 28.82 | 20.89 | 62.09 | 31.80 |
| **ConceptLM** | **15.20** | **28.52** | **19.68** | **55.05** | **29.61**↓ 6.74 |
| GPT2-1.5B | 14.99 | 27.89 | 20.72 | 55.12 | 29.68 |
| GPT2-PM | 14.94 | 27.95 | 21.00 | 56.15 | 30.01 |
| ContextLM | 14.60 | 27.05 | 17.46 | **44.62** | **25.93** |
| **ConceptLM** | **14.40** | **26.80** | **17.26** | 47.06 | 26.38↓ 3.30 |
| Pythia-70M | 18.27 | 57.01 | 142.01 | 973.59 | 297.72 |
| ContextLM | **14.96** | 43.64 | 71.45 | 440.77 | 142.71 |
| **ConceptLM** | 15.00 | **43.29** | **58.09** | **434.44** | **137.70**↓ 160.02 |
| Pythia-160M | 12.56 | 33.44 | 38.20 | 187.28 | 67.87 |
| ContextLM | 11.14 | 28.18 | 25.97 | 107.05 | 43.09 |
| **ConceptLM** | **10.93** | **27.01** | **20.01** | **79.32** | **34.23**↓ 33.54 |
| Pythia-410M | 8.88 | 20.11 | 10.85 | 31.53 | 17.84 |
| ContextLM | **8.67** | 19.50 | 10.15 | 24.95 | 15.82 |
| **ConceptLM** | 8.70 | **19.49** | **9.84** | **23.20** | **15.31**↓ 2.53 |
| Pythia-1B | 7.82 | 16.45 | 7.92 | 17.44 | 12.41 |
| ContextLM | 7.66 | 16.09 | **7.38** | 15.75 | 11.72 |
| **ConceptLM** | **7.58** | **15.84** | 7.68 | **14.69** | **11.45** ↓ 0.96 |
| Pythia-7B | 6.29 | 11.41 | 4.45 | 6.95 | 7.28 |
| **ConceptLM** | **5.99** | **10.95** | **4.31** | **6.92** | **7.04** ↓ 0.28 |

Table 1: Language modeling results for GPT-2 and Pythia models. O/P denotes OpenWebText results for GPT-2 models and Pile results for Pythia models.

The detailed results of language modeling tasks are shown at the top of Table 1, and the downstream tasks' performance is shown at the top of Table 2. We find that in language modeling tasks, our method achieves the best results except when testing Lambada-Standard on the XL-model. In downstream tasks, our model also consistently demonstrated the best average performance.

**Pythia Scaling Experiments:** For the Pythia series, we train ConceptLM from scratch. Results for the baseline and our strong baseline ContextLM, a concept-level model without explicit but implicit next concept prediction, are sourced from Dai et al. (2025). The language modeling performance of Pythia series is shown in Table 1, and the downstream performance is shown in Table 2.

We find that the average performance of our model on downstream tasks exceeds the baselines. For language modeling tasks, our model also generally outperforms the base model. While the 70M and 410M models perform slightly worse than ContextLM on the trained Pile dataset, our method demonstrates superior generalization in language modeling, outperforming ContextLM across several other datasets.

[1] All benchmarks except the Pile and OpenWebText are evaluated using the Language Model Evaluation Harness.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| GPT-124M | 27.3 | 20.7 | 41.6 | 17.9 | 49.6 | 60.4 | 27.5 | 69.5 | 27.4 | 38.0 |
| GPT-PM | 28.3 | 20.7 | 41.7 | 18.8 | 49.7 | 60.7 | 27.4 | 69.8 | 28.3 | 38.4 |
| ConceptLM | 29.2 | 20.3 | 43.8 | 19.3 | 52.9 | 60.7 | 27.6 | 69.5 | 28.4 | **39.1** |
| **ConceptLM** | 29.5 | 21.3 | 43.5 | 19.2 | 49.4 | 61.3 | 27.9 | 69.6 | 28.9 | 38.9↑ 0.9 |
| GPT-355M | 32.8 | 24.4 | 44.7 | 19.9 | 50.7 | 62.7 | 28.8 | 72.5 | 28.8 | 40.6 |
| GPT-PM | 35.2 | 25.3 | 44.6 | 20.4 | 52.6 | 62.4 | 29.1 | 73.1 | 29.8 | 41.4 |
| ContextLM | 36.8 | 26.7 | 46.7 | 19.5 | 50.9 | 62.9 | 29.8 | 74.2 | 29.3 | 41.9 |
| **ConceptLM** | 36.6 | 27.1 | 46.0 | 20.7 | 50.6 | 64.2 | 30.3 | 75.0 | 29.1 | **42.3**↑ 1.7 |
| GPT-774M | 37.2 | 27.4 | 47.0 | 20.2 | 53.6 | 64.6 | 30.3 | 74.9 | 29.8 | 42.8 |
| GPT-PM | 37.2 | 29.4 | 48.4 | 19.5 | 52.4 | 63.8 | 30.6 | 74.3 | 29.8 | 42.8 |
| ContextLM | 40.7 | 29.9 | 47.9 | 21.1 | 51.3 | 62.9 | 29.8 | 76.6 | 31.0 | 43.9 |
| **ConceptLM** | 41.1 | 30.6 | 48.5 | 19.9 | 52.1 | 65.3 | 31.4 | 77.0 | 30.0 | **44.0**↑ 1.2 |
| GPT-1.5B | 39.3 | 29.8 | 49.6 | 20.6 | 51.8 | 65.1 | 31.5 | 77.0 | 30.4 | 43.9 |
| GPT-PM | 38.7 | 29.5 | 49.2 | 20.0 | 51.2 | 64.8 | 31.5 | 77.0 | 29.9 | 43.5 |
| ContextLM | 41.9 | 32.3 | 49.9 | 21.4 | 51.9 | 66.1 | 32.3 | 77.7 | 31.2 | 45.0 |
| **ConceptLM** | 42.9 | 32.2 | 49.9 | 21.9 | 52.1 | 66.8 | 32.4 | 76.8 | 30.6 | **45.1**↑ 1.2 |
| Pythia-70M | 18.3 | 13.4 | 36.9 | 18.5 | 52.1 | 60.0 | 26.6 | 60.5 | 24.9 | 34.6 |
| ContextLM | 28.0 | 17.2 | 40.7 | 18.6 | 52.3 | 60.5 | 27.5 | 72.1 | 27.0 | **38.2** |
| **ConceptLM** | 29.3 | 17.5 | 40.2 | 17.8 | 50.8 | 59.8 | 27.5 | 74.3 | 26.2 | **38.2**↑ 3.6 |
| Pythia-160M | 32.7 | 21.5 | 43.8 | 19.5 | 53.4 | 61.5 | 28.5 | 74.3 | 27.9 | 40.3 |
| ContextLM | 37.2 | 25.6 | 45.0 | 19.5 | 52.0 | 62.9 | 29.2 | 77.6 | 28.7 | 42.0 |
| **ConceptLM** | 41.8 | 25.9 | 44.8 | 18.3 | 53.4 | 62.1 | 29.4 | 80.0 | 30.1 | **42.9**↑ 2.6 |
| Pythia-410M | 51.6 | 36.4 | 52.2 | 21.3 | 53.9 | 66.8 | 33.8 | 81.2 | 30.7 | 47.5 |
| ContextLM | 52.2 | 39.0 | 51.7 | 22.7 | 52.3 | 67.4 | 34.3 | 83.0 | 30.8 | 48.2 |
| **ConceptLM** | 53.0 | 39.7 | 52.0 | 22.1 | 54.2 | 66.6 | 34.3 | 82.2 | 30.2 | **48.3**↑ 0.8 |
| Pythia-1B | 55.9 | 42.0 | 56.8 | 24.2 | 52.5 | 70.5 | 37.7 | 83.3 | 32.7 | 50.6 |
| ContextLM | 57.8 | 43.7 | 55.3 | 25.9 | 55.0 | 70.1 | 37.6 | 86.1 | 32.1 | 51.5 |
| **ConceptLM** | 57.1 | 45.2 | 56.5 | 25.9 | 55.5 | 69.9 | 38.4 | 85.4 | 32.8 | **51.6**↑ 1.0 |
| Pythia-7B | 67.2 | 55.9 | 67.3 | 31.4 | 61.0 | 75.2 | 48.1 | 89.3 | 36.9 | 59.1 |
| **ConceptLM** | 68.7 | 57.3 | 68.2 | 32.9 | 61.0 | 75.6 | 49.0 | 91.6 | 37.1 | **60.2**↑ 1.1 |

Table 2: Evaluation of models across downstream tasks (Accuracy ↑) and language modeling (PPL ↓).

We suggest that the generalization performance of ConceptLM is attributed to the introduction of the concept vocabulary, which saves fundamental conceptual representations in a stable, discrete space after massive data training, thereby exhibiting significant generalization.

### 4.3 Scaling NCP to Trillion-Token Pretraining

We extend the OLMo-3 architecture (Team Olmo, 2025) with a deeper concept hierarchy, hierarchical residual routing, and an optimization recipe. The complete architecture of the scaled model is shown in Figure 7 in Appendix F, together with the residual and optimization details.

**Stage-1 training efficiency.** We train on 5.73T open-source Dolma 3 Mix tokens and compare token-level language-modeling loss with the OLMo-3-7B Stage-1 reference (Figure 4). The scaled model reaches the reference's final pretraining loss after consuming only 51.3% of its training tokens, a token-efficiency ratio of approximately $1/0.513 = 1.95$. Its final loss is lower by 0.091. This comparison concerns tokens needed to reach a loss threshold, not measured training time, and reflects the complete architecture and optimization recipe rather than the isolated effect of the NCP loss.

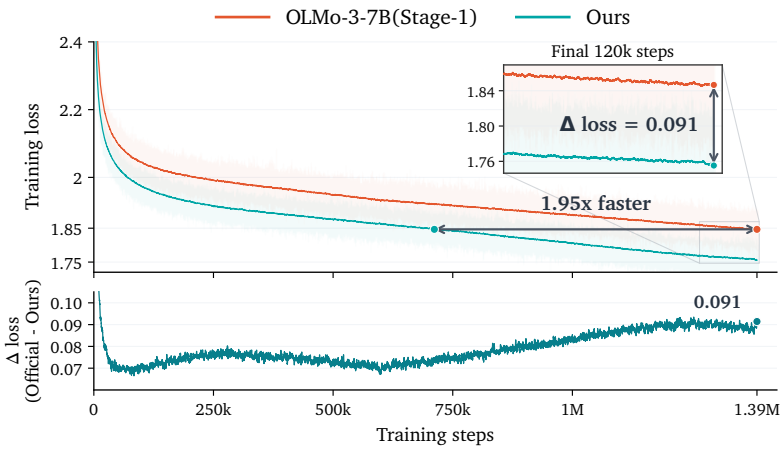



Figure 4: Stage-1 token-level pretraining loss. The original plot's 1.95x faster annotation denotes our training-token efficiency.

**Stage-1 downstream performance.** The full-suite downstream average increases from 46.59 to 49.04, a gain of 2.45 points. Table 3 highlights benchmarks with the largest absolute gains, including 8.60 points on PIQA, 5.99 on GSM8K, 5.29 on MultiPL-E MBPP, and 4.28 on HumanEval. Appendix F preserves every reported Stage-1 task score.

| Benchmark | OLMo-3-7B | NCP-8.9B | Δ |
|---|---|---|---|
| PIQA | 72.25 | 80.85 | +8.60 |
| GSM8K | 39.27 | 45.26 | +5.99 |
| MultiPL-E MBPP | 28.67 | 33.96 | +5.29 |
| HumanEval | 27.10 | 31.38 | +4.28 |
| GSM-Symbolic | 18.85 | 22.80 | +3.95 |
| SocialIQA | 65.35 | 69.24 | +3.89 |
| ARC-C | 77.99 | 81.57 | +3.58 |
| Full-suite average (26 tasks) | 46.59 | 49.04 | +2.45 |

Table 3: Stage-1 benchmarks with the largest absolute gains (higher is better).

## 5 Ablation Studies and Analysis

We analyze the scaled framework through OLMo-based pretraining experiments. Parameter- and computation-aligned comparisons and progressive component ablations use a shared 200B-token budget. Additional GPT-2/Pythia component comparisons, fixed-architecture objective ablations, and representation probes are provided in Appendix E.1.

### 5.1 Parameter- and Computation-Aligned Comparisons

We evaluate OLMo-based configurations over a shared 200B-token pretraining budget. The complete model has 32 token layers and eight concept layers operating at $k = 4$. At the block level, this corresponds to 40 token layers in parameters and 34 token layers in computation. We therefore compare it with 32-layer vanilla OLMo, a 34-layer computation-aligned model, and a 40-layer parameter-aligned model (Table 19).

The complete NCP model obtains lower training loss than the computation-aligned baseline and approaches the parameter-aligned baseline while using approximately $34/40 = 85\%$ of its computation (Figure 5). This supports efficient use of the added concept layers at the evaluated budget.

### 5.2 Concept Module and Training Objective

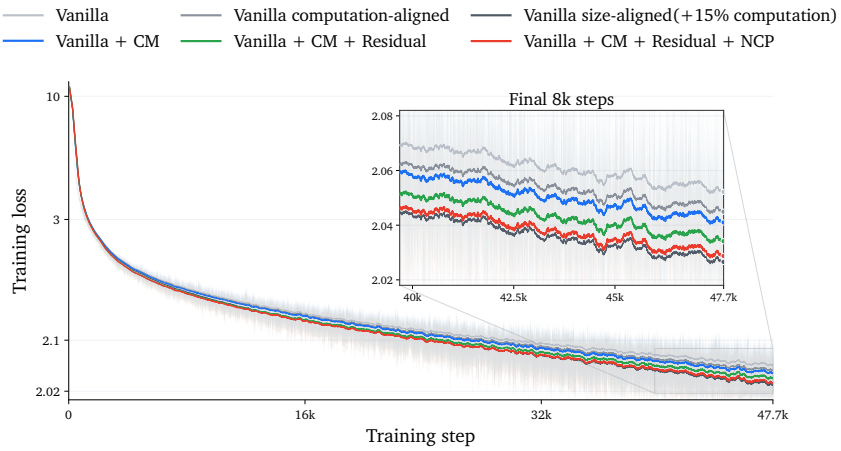



Figure 5: OLMo-3-7B based pretraining controls over 200B tokens under a shared recipe. The component progression is Vanilla, +CM (Concept Module), +CM+Residual, and +CM+Residual+NCP. The two other curves provide parameter- and computation-aligned references.

We progressively add concept computation, hierarchical residual routing, and the NCP objective to the vanilla model under the same 200B-token training recipe (Figure 5). Each addition reduces the observed training loss. In particular, comparing +CM+Residual with +CM+Residual+NCP holds the scaled architecture fixed and isolates the contribution of the additional predictive objective within this configuration. Thus, explicit concept prediction further improves the enhanced architecture. The progressive study evaluates our scaled design, and Appendix E.1.2 separately examines NCP and VQ objectives with the basic Pythia architecture held fixed.

| Model | LM loss ↓ | Training FLOPs ($10^{20}$) |
|---|---|---|
| OLMo-3-3B | 2.3894 | 8.194 |
| OLMo + Residual | 2.3707 | 8.194 |
| OLMo + Residual + MTP | 2.3610 | 9.468 |
| NCP | 2.3571 | 8.148 |
| NCP + MTP | **2.3497** | 9.422 |

Table 4: Residual-aligned 3B study. NCP improves over the residual baseline and remains complementary to MTP. These runs do not share the 8.9B model's total token budget. FLOPs use the analytical accounting of the experiment.

### 5.3 Complementarity with multi-token prediction

We compare NCP with multi-token prediction (MTP) (Gloeckle et al., 2024) in a separate residual-aligned 3B study. Table 4 reports primary LM loss, excluding auxiliary losses, averaged over the final third of training. NCP without MTP has lower late-training loss than the residual baseline with MTP. Adding MTP further improves NCP, indicating complementary signals in this setting. Comparing the two MTP variants also favors NCP at slightly lower analytical FLOPs.

## 6 Conclusion

In this paper, we propose Next Concept Prediction (NCP), a novel and more challenging pretraining objective built upon a discrete concept-level latent space. By leveraging VQ to construct this space and integrating it with standard Next Token Prediction (NTP), we develop the ConceptLM framework. Our results demonstrate that ConceptLM exhibits superior performance on language modeling and downstream tasks compared with standard token-level models and strong baselines. Moreover, we show the applicability, robustness, and low-cost nature of our method. Finally, we provide empirical evidence of the model's scalability across different parameter scales and backbones. These findings suggest that transitioning the supervision space for language modeling is a promising path for building the next generation of LLMs.

## Ethical Considerations

ConceptLM is a framework that incorporates explicit high-level supervision into language models, where the proposed NCP (Next Concept Prediction) pretraining task enables superior scalability and generalization. This shifts language modeling from low-level token modeling to explicit high-level supervision, representing a paradigm shift that contributes to the advancement of future powerful language models.

## AI Use Statement

Generative AI tools assisted with polishing the paper. The authors are responsible for experimental validation and the final content.

## References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. URL https://arxiv.org/abs/2006.11477.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL https://arxiv.org/abs/2404.08471.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021. URL https://arxiv.org/abs/2106.01345.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Beiya Dai, Yuliang Liu, Daozheng Xue, Qipeng Guo, Kai Chen, Xinbing Wang, Bowen Zhou, and Zhouhan Lin. Context-level language modeling by learning predictive context embeddings, 2025. URL https://arxiv.org/abs/2510.20280.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step, 2024. URL https://arxiv.org/abs/2405.14838.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction, 2024. URL https://arxiv.org/abs/2404.19737.

Aaron Grattafiori and et.al. Abhimanyu Dubey. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. doi: 10.1109/MASSP.1984.1162229.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units, 2021. URL https://arxiv.org/abs/2106.07447.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving, 2023. URL https://arxiv.org/abs/2309.17080.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations, 2017. URL https://arxiv.org/abs/1704.04683.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. 2022.

Jingyu Liu and Ce Zhang. Hamburger: Accelerating llm inference via token smashing, 2025. URL https://arxiv.org/abs/2505.20438.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, et al. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pp. 142–150, USA, 2011. Association for Computational Linguistics. ISBN 9781932432879.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1559–1571, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.117. URL https://aclanthology.org/2022.findings-naacl.117/.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Xingwei Qu, Shaowen Wang, Zihao Huang, Kai Hua, Fan Yin, Rui-Jie Zhu, Jundong Zhou, Qiyang Min, Zihao Wang, Yizhi Li, Tianyu Zhang, He Xing, Zheng Zhang, Yuxuan Song, Tianyu Zheng, Zhiyuan Zeng, Chenghua Lin, Ge Zhang, and Wenhao Huang. Dynamic large concept models: Latent reasoning in an adaptive semantic space, 2026. URL https://arxiv.org/abs/2512.24617.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Chenze Shao, Fandong Meng, and Jie Zhou. Beyond next token prediction: Patch-level training for large language models, 2025. URL https://arxiv.org/abs/2407.12665.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning, 2025. URL https://arxiv.org/abs/2502.03275.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. Llm pretraining with continuous concepts, 2025. URL https://arxiv.org/abs/2502.08524.

LCM team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, Safiyyah Saleem, and Holger Schwenk. Large concept models: Language modeling in a sentence representation space, 2024. URL https://arxiv.org/abs/2412.08821.

Team Olmo. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction, 2024. URL https://arxiv.org/abs/2404.02905.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions, 2017. URL https://arxiv.org/abs/1707.06209.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. MUDDFormer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections, 2025. URL https://arxiv.org/abs/2502.12170.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui

Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers, 2023. URL https://arxiv.org/abs/2305.07185.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016. URL https://arxiv.org/abs/1509.01626.

Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22968–22977, October 2025.

# A Experimental Setup

## A.1 Model and Data Summary

Table 5 summarizes the model backbones, training stages, datasets, and token budgets used in our experiments.

| Model Backbone | Parameters | Training Stage | Training Dataset | Token Budget |
|---|---|---|---|---|
| GPT-2 | 160M-1.5B | Pretraining | OpenWebText | 8B |
| Pythia | 70M-410M | Pretraining | The Pile | 300B |
| Olmo-3 | 7B | Pretraining | Dolma-3-mix | 5.73T |

Table 5: Summary of experimental settings.

## A.2 ConceptLM Configuration

Table 6 lists the main ConceptLM configuration, including insertion depth, number of concept layers, chunk size, and concept vocabulary parameters.

| Model Backbone | Insert Depth | Concept Layers | Chunk Size | $S$ | $N$ |
|---|---|---|---|---|---|
| GPT-2 | before the first/second layer | 2 | 4 | head number | 64 |
| Pythia | before the second layer | 2 | 4 | head number | 64 |
| Olmo | in the middle of the model | 8 | 4 | head number | 128 |

Table 6: Summary of ConceptLM settings in the main experiments.

**Insert Depth Discussion.** To ensure that more decoder layers benefit from multi-level information while maintaining token-level feature mixing before the concept layer, we default to inserting concept layers after the first token layer of the backbone. In this configuration, the first layer acts as the token-level encoder, while the remaining $L - 1$ layers serve as the token-level decoder.

However, given that the GPT-2 model was pretrained on a relatively small dataset (less than 10B tokens), its performance is highly sensitive to architectural depth. To ensure a fair comparison with our strong baseline, ContextLM, we conducted additional experiments on GPT-2 by inserting the concept layer before the first token-level layer. Results for the configuration where the concept layer is inserted after the first layer are reported in Appendix D.2.

Furthermore, we conducted extensive ablation studies to analyze the impact of the insert depth. Our results indicate that placing the concept layers at either the very beginning (Layer 0) or after the first layer (Layer 1) is suboptimal. Instead, the model achieves the best performance when the layer is positioned at three-quarters of the backbone depth. We hypothesize that this superiority stems from the backpropagation of concept-level supervision into the Encoder layers: a sufficiently deep Encoder provides more refined representations, thereby facilitating more accurate concept prediction. Detailed results are provided in Appendix E.2.

## A.3 Training Configurations

Tables 7 and 8 report the training configurations for the GPT-2 and Pythia experiments. Within each model family and size, all compared variants use the same training setting.

# B Evaluation Tasks

Table 9 summarizes the evaluated benchmarks and their associated capabilities.

| Model Size | Learning Rate | Batch Size | Training Length |
|---|---|---|---|
| GPT-Base (124M) | 1e-3 | 512 | 1024 |
| GPT-Medium (355M) | 8e-4 | 512 | 1024 |
| GPT-Large (774M) | 6e-4 | 512 | 1024 |
| GPT-XL (1.5B) | 4e-4 | 512 | 1024 |

Table 7: GPT-2 training configuration.

| Model Size | Learning Rate | Batch Size | Training Length |
|---|---|---|---|
| Pythia-70M | 1e-3 | 1024 | 2048 |
| Pythia-160M | 6e-4 | 1024 | 2048 |
| Pythia-410M | 4e-4 | 1024 | 2048 |
| Pythia-1B | 2.5e-4 | 1024 | 2048 |
| Pythia-7B | 1.2e-4 | 1024 | 2048 |

Table 8: Pythia training configuration.

# C Efficiency and Scaling Analysis

## C.1 Scaling Performance

Table 10 shows that introducing concept-level modeling improves scaling behavior. For standard token-level models, simply stacking additional token layers does not yield stable gains, and performance even degrades when the depth reaches 22 token-level layers. This phenomenon can be attributed to depth-width imbalance or representation bottlenecks, where the limited dimension cannot effectively support the increased abstraction required by excessive depth.

By contrast, incorporating concept-level modeling with a certain number of token layers helps ConceptLM overcome these scaling limitations. With concept-level layers operating on a denser semantic space, ConceptLM exhibits a widening PPL.Avg gap as depth increases, underscoring its robust scaling behavior under a fixed-width configuration.

## C.2 Computational Complexity

Let $T$ be the input sequence length, $d$ the hidden dimension, and $k$ the chunk size. The training complexity of a token-level layer is $O(T^2d + Td^2)$, while a concept-level layer operating on $T/k$ positions has complexity $O((T/k)^2d + (T/k)d^2)$. Thus, sequence compression reduces the attention-score computation by a factor of $k^2$ and the position-wise computation by a factor of $k$ within the concept-level layers. This advantage becomes more substantial as the sequence length $T$ increases.

| **Capability** | **Benchmarks / Datasets** |
|---|---|
| Long-range dependency & context consistency | Lambada (OpenAI / Standard) |
| Commonsense reasoning | ARC-Easy, ARC-Challenge, WinoGrande, PIQA, HellaSwag |
| Scientific knowledge & reasoning | SciQ |
| Multi-sentence / paragraph comprehension | RACE |
| General language modeling | The Pile, WikiText, OpenWebText |

Table 9: Evaluated benchmarks with capabilities.

| Model | PPL.Avg ↓ | ACC.Avg ↑ |
|---|---|---|
| +2 token-layers | 71.35 | 40.13 |
| **+2 concept-layers** | **68.13**+3.22 | **40.26** +0.13 |
| +6 token-layers | 62.25 | 40.09 |
| **+6 concept-layers** | **54.34**+7.91 | **40.91**+0.82 |
| +10 token-layers | 75.20 | 40.18 |
| **+10 concept-layers** | **58.21**+16.99 | **40.38**+0.20 |

Table 10: Comparison of model scaling with or without concept layers.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| MTP | 32.1 | 20.5 | 40.2 | 17.6 | 49.4 | 60.2 | 28.0 | 73.3 | 27.0 | 38.7 |
| ConceptLM | 34.0 | 21.8 | 43.5 | 18.8 | 53.0 | 61.3 | 28.3 | 73.3 | 28.6 | **40.3** |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| MTP | 14.83 | 34.41 | 37.36 | 222.18 | — | 77.20 |
| ConceptLM | 12.64 | 33.45 | 36.38 | 190.06 | — | **68.13** |

Table 11: MTP baseline. ConceptLM uses the NTP, NCP, and VQ objectives.

# D Additional Experimental Results

## D.1 MTP Baseline

We train an MTP model as an additional baseline. Empirical evidence, including the original MTP paper, suggests that MTP models do not yield superior performance at scales below 3B parameters, larger than our largest 1.5B pretrained model. Consequently, rather than categorizing it as a strong baseline in the main body, we train a small MTP variant and report the result in Table 11.

## D.2 GPT-2 Results with L1 Setting

Table 12 reports the GPT-2 experiments in which concept-level layers are inserted after the first token-level layer.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| GPT-124M | 27.3 | 20.7 | 41.6 | 17.9 | 49.6 | 60.4 | 27.5 | 69.5 | 27.4 | 38.0 |
| GPT-PM | 28.3 | 20.7 | 41.7 | 18.8 | 49.7 | 60.7 | 27.4 | 69.8 | 28.3 | 38.4 |
| ContextLM | 29.2 | 20.3 | 43.8 | 19.3 | 52.9 | 60.7 | 27.6 | 69.5 | 28.4 | **39.1** |
| **ConceptLM-L1** | 28.9 | 20.7 | 41.9 | 18.0 | 50.0 | 60.2 | 27.8 | 70.8 | 30.1 | 38.7+0.7 |
| **ConceptLM** | 29.5 | 21.3 | 43.5 | 19.2 | 49.4 | 61.3 | 27.9 | 69.6 | 28.9 | 38.9+0.9 |
| GPT-355M | 32.8 | 24.4 | 44.7 | 19.9 | 50.7 | 62.7 | 28.8 | 72.5 | 28.8 | 40.6 |
| GPT-PM | 35.2 | 25.3 | 44.6 | 20.4 | 52.6 | 62.4 | 29.1 | 73.1 | 29.8 | 41.4 |
| ContextLM | 36.8 | 26.7 | 46.7 | 19.5 | 50.9 | 62.9 | 29.8 | 74.2 | 29.3 | 41.9 |
| **ConceptLM-L1** | 34.1 | 25.3 | 46.1 | 20.1 | 50.4 | 63.3 | 29.7 | 76.0 | 28.7 | 41.5+0.9 |
| **ConceptLM** | 36.6 | 27.1 | 46.0 | 20.7 | 50.6 | 64.2 | 30.3 | 75.0 | 29.1 | **42.3**+1.7 |
| GPT-774M | 37.2 | 27.4 | 47.0 | 20.2 | 53.6 | 64.6 | 30.3 | 74.9 | 29.8 | 42.8 |
| GPT-PM | 37.2 | 29.4 | 48.4 | 19.5 | 52.4 | 63.8 | 30.6 | 74.3 | 29.8 | 42.8 |
| ContextLM | 40.7 | 29.9 | 47.9 | 21.1 | 51.3 | 62.9 | 29.8 | 76.6 | 31.0 | 43.9 |
| **ConceptLM-L1** | 39.4 | 29.6 | 49.1 | 21.5 | 50.8 | 64.3 | 31.3 | 75.1 | 30.5 | 43.5+0.7 |
| **ConceptLM** | 41.1 | 30.6 | 48.5 | 19.9 | 52.1 | 65.3 | 31.4 | 77.0 | 30.0 | **44.0**+1.2 |
| GPT-1.5B | 39.3 | 29.8 | 49.6 | 20.6 | 51.8 | 65.1 | 31.5 | 77.0 | 30.4 | 43.9 |
| GPT-PM | 38.7 | 29.5 | 49.2 | 20.0 | 51.2 | 64.8 | 31.5 | 77.0 | 29.9 | 43.5 |
| ContextLM | 41.9 | 32.3 | 49.9 | 21.4 | 51.9 | 66.1 | 32.3 | 77.7 | 31.2 | 45.0 |
| **ConceptLM-L1** | 41.7 | 32.2 | 49.7 | 21.0 | 52.0 | 65.4 | 31.9 | 77.9 | 30.7 | 44.7+0.8 |
| **ConceptLM** | 42.9 | 32.2 | 49.9 | 21.9 | 52.1 | 66.8 | 32.4 | 76.8 | 30.6 | **45.1**+1.2 |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| GPT-124M | 22.04 | 43.99 | 69.29 | 272.43 | — | 101.94 |
| GPT-PM | 21.22 | 42.15 | 60.05 | 273.12 | — | 99.14 |
| ContextLM-L1 | 20.68 | 41.45 | 55.22 | 231.41 | — | 87.19 |
| **ConceptLM-L1** | 21.19 | 42.13 | 56.84 | 230.41 | — | 87.64+14.30 |
| **ConceptLM** | **20.56** | **40.86** | **51.83** | **225.74** | — | **84.75**+17.19 |
| GPT-355M | 17.97 | 34.24 | 35.54 | 122.54 | — | 52.57 |
| GPT-PM | 17.67 | 33.59 | 32.91 | 126.03 | — | 52.55 |
| ContextLM | 17.03 | 32.08 | 28.06 | 95.06 | — | 43.06 |
| **ConceptLM-L1** | 17.14 | 32.42 | 31.29 | 100.50 | — | 43.34+9.23 |
| **ConceptLM** | **16.81** | **31.53** | **26.96** | **86.75** | — | **40.51**+12.06 |
| GPT-774M | 16.01 | 30.03 | 24.58 | 74.76 | — | 36.35 |
| GPT-PM | 15.89 | 29.71 | 23.50 | 62.45 | — | 32.89 |
| ContextLM | 15.41 | 28.82 | 20.89 | 62.09 | — | 31.80 |
| **ConceptLM-L1** | 15.47 | 29.66 | 21.55 | 61.52 | — | 32.05+4.30 |
| **ConceptLM** | **15.20** | **28.52** | **19.68** | **55.05** | — | **29.61**+6.74 |
| GPT-1.5B | 14.99 | 27.89 | 20.72 | 55.12 | — | 29.68 |
| GPT-PM | 14.94 | 27.95 | 21.00 | 56.15 | — | 30.01 |
| ContextLM | 14.60 | 27.05 | 17.46 | **44.62** | — | **25.93** |
| **ConceptLM-L1** | 14.51 | 26.83 | 18.47 | 49.09 | — | 27.23+2.45 |
| **ConceptLM** | **14.40** | **26.80** | **17.26** | 47.06 | — | 26.38+3.30 |

Table 12: Evaluation of models across downstream tasks (Accuracy ↑) and language modeling (PPL ↓). ConceptLM-GPT is denoted as ConceptLM. ConceptLM-L1 denotes that we insert the concept-level module after the first layer. ConceptLMdenotes that we insert the concept-level module before the first layer.

# E Ablation Studies

## E.1 Additional GPT-2 and Pythia Analysis

The following studies analyze the basic framework without the additional hierarchical residual routing. Unless otherwise specified, objective ablations use a Pythia-160M backbone trained from scratch on 30B tokens. The representation probes use ConceptLM-Pythia-7B.

### E.1.1 Concept-level Modeling Modules

To evaluate the effectiveness of concept layers and concept vocabulary separately, we compare three configurations under the same experimental settings, as reported in Table 1 and Table 2. Specifically, the comparison between the PM (Parameter-Matching) models and ConceptLM models highlights the superiority of concept-level modeling in enhancing model capabilities. Furthermore, the performance gap between ContextLM and ConceptLM on the Pythia experiments underscores the generalization advantages of explicitly incorporating a concept vocabulary compared to implicit concept modeling.

### E.1.2 Training Objectives

| Model | PPL.Avg ↓ | ACC.Avg ↑ |
|---|---|---|
| $\mathcal{L}_{\text{NTP}}$ | 69.28 | 39.54 |
| $\mathcal{L}_{\text{NTP}}$ + $\mathcal{L}_{\text{VQ}}$ | 75.64 | 39.41 |
| $\mathcal{L}_{\text{NTP}}$ + $\mathcal{L}_{\text{NCP}}$ | 76.30 | 39.62 |
| $\mathcal{L}_{\text{NTP}}$ + $\mathcal{L}_{\text{NCP}}$ + $\mathcal{L}_{\text{VQ}}$ | **68.13** | **40.26** |

Table 13: Ablation of training objective.

We conduct an ablation study by systematically removing $\mathcal{L}_{\text{NCP}}$ and $\mathcal{L}_{\text{VQ}}$ to evaluate the contribution of each loss, while keeping the model architecture unchanged. As shown in Table 13, using $\mathcal{L}_{\text{NTP}}$ alone provides the model with solid foundational capabilities. Using either $\mathcal{L}_{\text{VQ}}$ or $\mathcal{L}_{\text{NCP}}$ independently leads to a performance drop. This suggests that without quantization constraints, the vocabulary fails to learn robust concept representations; conversely, without the supervision of concept prediction, the model cannot effectively predict useful discrete concepts. Optimal performance is only achieved when objectives are combined, achieving the lowest reported mean perplexity (68.13) and highest downstream average (40.26), compared with 69.28 and 39.54 for NTP alone.

### E.1.3 Concept Semantic Analysis

| Dataset | Dec. Acc. | Pool Acc. |
|---|---|---|
| AG News Zhang et al. (2016) | 87.13 | 89.03 |
| IMDB Maas et al. (2011) | 78.15 | 77.46 |
| DBPedia-14 Zhang et al. (2016) | 95.44 | 96.42 |
| Yahoo Topics Zhang et al. (2016) | 65.97 | 63.62 |
| Avg | 81.67 | 81.63 |

Table 14: Classification performance of fixed concept representations. Dec. denotes decoded fixed concepts, Pool denotes pooled hidden representations.

We further investigate whether the learned fixed concepts encode semantics on ConceptLM-Pythia-7B. We evaluate concept representations on four standard probing datasets: AG News, IMDB, DBPedia-14, and Yahoo Topics, comparing two types of representations: predicted concepts and pooled token-level hidden representations. As shown in Table 14, decoded fixed concepts achieve strong classification performance. This result indicates that the predicted concepts are more semantically separable than the pooled token representations, suggesting that the model effectively learns to predict meaningful high-level abstractions.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| ConceptLM-L0 | 34.2 | 22.4 | 45.2 | 19.0 | 51.0 | 61.5 | 28.1 | 75.1 | 27.9 | **40.5** |
| ConceptLM-L1 | 34.0 | 21.8 | 43.5 | 18.8 | 53.0 | 61.3 | 28.3 | 73.3 | 28.6 | 40.3 |
| ConceptLM-L3 | 32.2 | 21.6 | 42.0 | 19.7 | 51.1 | 61.4 | 28.1 | 71.9 | 28.8 | 39.6 |
| ConceptLM-L5 | 33.9 | 22.7 | 43.5 | 19.0 | 51.9 | 61.3 | 28.3 | 73.0 | 29.0 | 40.3 |
| ConceptLM-L7 | 34.0 | 21.7 | 42.9 | 17.5 | 51.6 | 62.5 | 28.2 | 75.2 | 28.7 | 40.3 |
| ConceptLM-L9 | 34.9 | 24.2 | 43.1 | 19.7 | 50.3 | 61.9 | 27.8 | 74.2 | 27.9 | 40.4 |
| ConceptLM-L11 | 33.9 | 22.3 | 43.3 | 18.7 | 53.6 | 61.5 | 28.3 | 73.8 | 27.9 | 40.3 |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| ConceptLM-L0 | 12.52 | 33.74 | 37.46 | 166.29 | — | 63.25 |
| ConceptLM-L1 | 12.64 | 33.45 | 36.38 | 190.06 | — | 68.13 |
| ConceptLM-L3 | 12.65 | 33.70 | 38.94 | 163.07 | — | 62.09 |
| ConceptLM-L5 | 12.66 | 33.62 | 38.09 | 158.20 | — | 60.64 |
| ConceptLM-L7 | 12.62 | 33.49 | 35.86 | 175.37 | — | 64.34 |
| ConceptLM-L9 | 12.64 | 33.72 | 35.49 | 138.28 | — | **55.04** |
| ConceptLM-L11 | 18.67 | 32.26 | 36.96 | 150.88 | — | 59.79 |

Table 15: Insertion depth ablation. Evaluation of the Pythia-160M backbone trained on 30B Pile tokens; L0 denotes that there are 0 token-level layers before the concept-level module.

### E.2 Insert Depth Ablation

We insert the concept-level layers at L0, L1, L3, L5, L7, L9, and L11. Results are shown in Table 15. We find that inserting the concept layers at the three-quarters (3/4) depth of the model yields superior generalization performance.

### E.3 VQ Module Setting

In our early experiments, we employ a vanilla VQ module without the SimVQ trick. We observe that the codebooks frequently collapse when the concept layers are inserted before the first layer. Training stability is achieved only by applying complex transformations to the codebook through non-linear activation functions. We therefore keep the two-MLP-layer and ReLU activation setting.

We hypothesize that this collapse occurs because a simple linear projection, or no projection, lacks the capacity to balance the complex representations with the discrete concept vocabulary, causing most inputs to be mapped to a few specific codebook entries. The introduction of non-linear activations redistributes the representations, preventing the model from converging to a few codes and significantly increasing codebook utilization.

Across all experimental configurations in this paper, the VQ codebook utilization consistently reaches nearly 100%, demonstrating that our design effectively mitigates the common issue of codebook collapse. We provide an early experiment to test codebook usage even with a large codebook setting $S = 12, N = 1024$.

### E.4 Chunk Size Ablation

We provide experimental results for the chunk size ablation in Table 16. A larger chunk size reduces layer computational cost, showing a clear trade-off between computational cost and model capability.

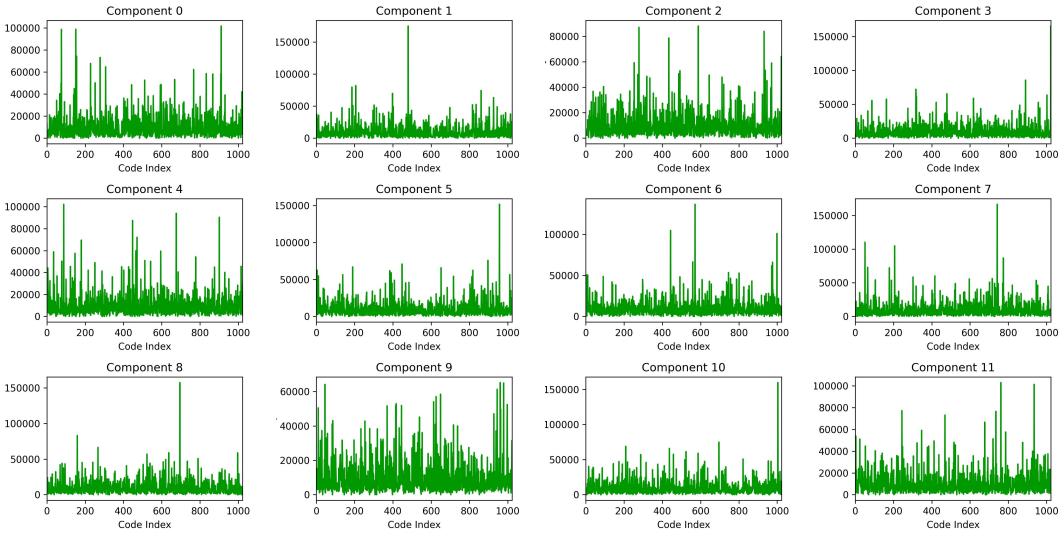



Figure 6: Codebook usage. The figure shows the usage frequency of each code entry across 12 codebooks, each containing 1,024 entries, evaluated on the OpenWebText test set.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| Chunk-2 | 35.6 | 22.6 | 44.7 | 19.5 | 53.0 | 60.8 | 28.1 | 75.9 | 28.2 | **40.7** |
| Chunk-4 | 34.0 | 21.8 | 43.5 | 18.8 | 53.0 | 61.3 | 28.3 | 73.3 | 28.6 | 40.3 |
| Chunk-8 | 33.7 | 22.5 | 43.5 | 19.2 | 52.3 | 61.4 | 27.3 | 73.0 | 28.6 | 40.2 |
| Chunk-16 | 33.4 | 21.9 | 42.8 | 19.7 | 50.7 | 60.8 | 27.0 | 74.2 | 28.6 | 39.8 |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| Chunk-2 | 12.59 | 33.52 | 34.89 | 158.19 | — | **59.80** |
| Chunk-4 | 12.64 | 33.45 | 36.38 | 190.06 | — | 68.13 |
| Chunk-8 | 12.67 | 33.82 | 37.35 | 179.41 | — | 65.81 |
| Chunk-16 | 12.73 | 34.09 | 38.73 | 193.26 | — | 69.70 |

Table 16: Chunk size ablation. Evaluation of the Pythia-160M backbone trained on 30B Pile tokens; Chunk-2 denotes chunk size = 2 ($k = 2$).

### E.5 Loss Ablation

We provide detailed loss ablation results for Appendix E.1.2 in Table 17.

### E.6 Detailed Scaling Analysis

We provide detailed scaling analysis results for Section 4.2 in Table 18.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| NTP | 32.9 | 20.7 | 43.4 | 18.0 | 50.7 | 61.3 | 28.0 | 72.3 | 27.0 | 39.6 |
| NTP + VQ | 34.4 | 20.5 | 43.4 | 18.0 | 49.1 | 61.4 | 28.0 | 72.3 | 27.6 | 39.8 |
| NTP + NCP | 34.0 | 20.5 | 43.1 | 18.3 | 49.4 | 61.3 | 28.1 | 73.2 | 27.9 | 39.5 |
| NTP + NCP + VQ | 34.0 | 21.8 | 43.5 | 18.8 | 53.0 | 61.3 | 28.3 | 73.3 | 28.6 | **40.3** |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| NTP | 12.64 | 34.30 | 37.31 | 192.87 | — | 69.28 |
| NTP + VQ | 12.62 | 33.66 | 37.60 | 218.69 | — | 75.64 |
| NTP + NCP | 13.21 | 35.95 | 41.35 | 214.70 | — | 76.30 |
| NTP + NCP + VQ | 12.64 | 33.45 | 36.38 | 190.06 | — | **68.13** |

Table 17: Detailed loss ablation results.

| **Model** | Lambada OpenAI | Lambada Std. | ARC-E | ARC-C | Wino Gd. | PIQA | Hella-Swag | SciQ | RACE | **Avg Acc ↑** |
|---|---|---|---|---|---|---|---|---|---|---|
| *Zero-shot Evaluation (Accuracy %)* | | | | | | | | | | |
| +2 token | 33.7 | 21.0 | 44.5 | 20.1 | 50.7 | 61.4 | 27.9 | 74.7 | 27.4 | 40.1 |
| +2 concept | 34.0 | 21.8 | 43.5 | 18.8 | 53.0 | 61.3 | 28.3 | 73.3 | 28.6 | 40.3 |
| +6 token | 34.7 | 22.5 | 44.8 | 18.8 | 52.1 | 60.2 | 27.6 | 72.3 | 28.6 | 40.1 |
| +6 concept | 33.9 | 23.3 | 43.0 | 19.7 | 54.1 | 61.0 | 28.4 | 76.0 | 28.6 | 40.9 |
| +10 token | 33.0 | 20.7 | 44.0 | 19.8 | 52.0 | 61.9 | 28.1 | 74.7 | 27.0 | 40.2 |
| +10 concept | 33.9 | 22.3 | 43.3 | 18.7 | 53.6 | 61.5 | 28.3 | 73.8 | 27.9 | 40.4 |

| **Model** | OWT | Wikitext | Lambada OpenAI | Lambada Std | — | **Avg PPL ↓** |
|---|---|---|---|---|---|---|
| *Language Modeling (Perplexity)* | | | | | | |
| +2 token | 12.77 | 34.02 | 37.94 | 200.67 | — | 59.80 |
| +2 concept | 12.64 | 33.45 | 36.38 | 190.06 | — | 68.13 |
| +6 token | 12.74 | 34.06 | 37.31 | 176.90 | — | 65.25 |
| +6 concept | 12.45 | 33.06 | 37.83 | 134.48 | — | 54.34 |
| +10 token | 12.73 | 34.06 | 40.69 | 213.06 | — | 75.20 |
| +10 concept | 12.32 | 33.26 | 36.96 | 150.88 | — | 58.21 |

Table 18: Detailed scaling analysis results.

# F OLMo-Based Stage-1 Pretraining: Full Results

We provide the configuration, evaluation protocol, and complete results for the pretrained Stage-1 checkpoints. Unless otherwise specified, the results correspond to the 5.73T-token run.

## F.1 Complete architecture and optimization

Figure 7 presents the complete NCP-ArchPreview model, including the token-level hierarchy, Concept Module, segmented concept vocabulary, and causally aligned concept-to-token pathway. The scaled model uses a 16/8/16 Encoder/Concept/Decoder depth allocation, hidden size 4,096, FFN size 11,008, 32 attention heads, $k = 4$, 32 codebooks with 128 entries per codebook, and context length 8,192. It retains layer-wise Q/K normalization.

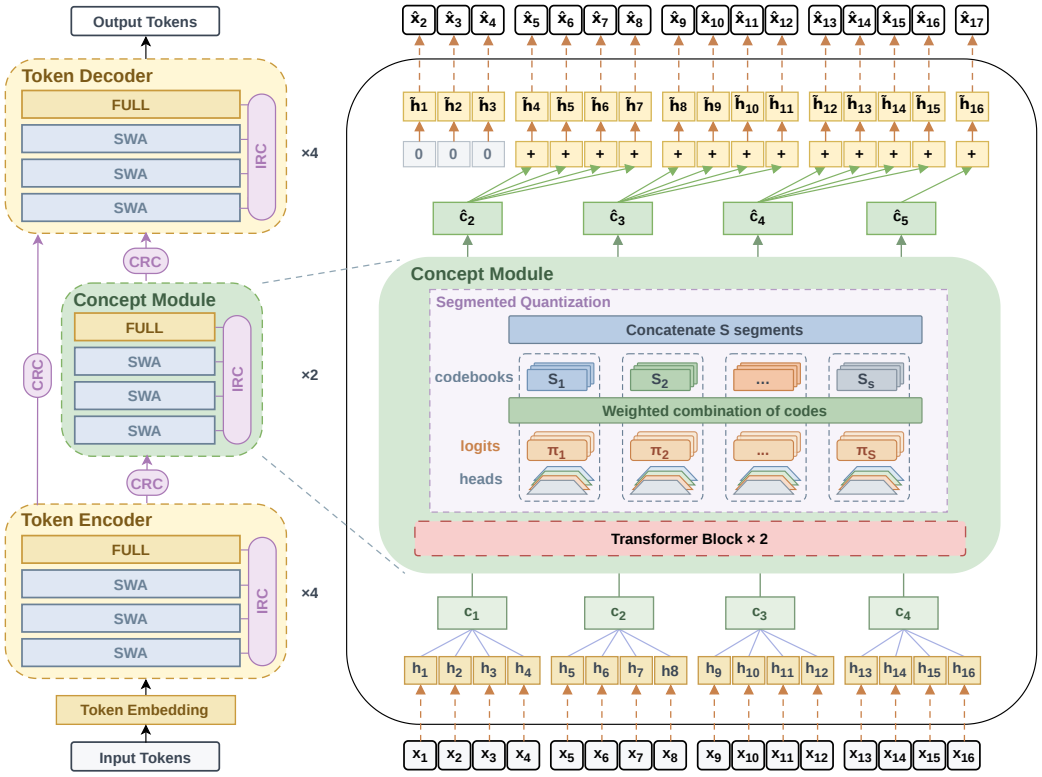



Figure 7: Complete architecture of the OLMo-based scaled model, reproduced from the NCP-ArchPreview model diagram. The left panel shows the Encoder–Concept Module–Decoder hierarchy and IRC/CRC routing; the right panel expands concept prediction, segmented codebooks, and the causal alignment with token prediction. The left panel specifies the full 16/8/16 depth allocation; the right panel is a schematic of the concept pathway.

**Hierarchical residual connections.** Intra-module residual connections (IRC), inspired by MUDDFormer (Xiao et al., 2025), mix states from different depths using hidden-state-dependent weights. The weights are not normalized and are initialized to select the most recent residual-updated state, recovering the standard residual connection at initialization. Cross-module residual connections (CRC) transfer aligned representations from Encoder to Concept Module, Encoder to Decoder, and Concept Module to Decoder. Source representations are aligned by chunking or repetition, combined with softmax-normalized depth weights, and added to the target using learned diagonal scaling. The Concept Module-to-Decoder connection follows the causal shift of the concept pathway.

**Optimization recipe.** We adopt Moonlight Muon (Liu et al., 2025) for matrix-valued parameters and AdamW for embeddings, biases, and the remaining parameters, with a learning rate of $6 \times 10^{-5}$ and an OLMo-style cosine schedule. These changes are intended to improve overall model performance. The 200B-token controls in Section 5.1 share the same training recipe and therefore verify architecture and objective choices.

### F.2 Parameter- and computation-aligned configurations

| Configuration | Token layers | Concept layers | $P/P_{\text{blk}}$ / $F/F_{\text{blk}}$ |
|---|---|---|---|
| Vanilla | 32 | 0 | 32 / 32 |
| Computation-aligned | 34 | 0 | 34 / 34 |
| Parameter-aligned | 40 | 0 | 40 / 40 |
| NCP | 32 | 8 | 40 / 34 |

Table 19: Approximate parameter and computation controls, in units of one standard OLMo block. Small interface costs and non-contraction overheads are excluded.

### F.3 Evaluation and complete results

The overall downstream score averages 26 constituent results: four MMLU domains and MMLU-Pro, four mathematics scores, six code scores, two MC-STEM scores, three MC-Non-STEM scores, and six GenQA scores. The aggregate MMLU row is displayed but not counted a second time; category averages are likewise excluded. BPB scores form a separate likelihood evaluation, with lower values better. Their average is taken across ten benchmark-level results; the MT-MBPP language subsets are aggregated first. The complete retained scores are provided below rather than selecting only improved tasks.

Table 20: Complete retained Stage-1 accuracy and BPB results. Accuracy scores are percentages. The aggregate MMLU score is excluded from the 26-task accuracy average; BPB is averaged separately.

| Benchmark | OLMo-3-7B | NCP-8.9B |
|---|---|---|
| **Accuracy** (↑) | | |
| MMLU-Humanities | 64.73 | 68.28 |
| MMLU-Social Sciences | 71.18 | 73.97 |
| MMLU-STEM | 53.04 | 55.53 |
| MMLU-Other | 64.00 | 65.63 |
| MMLU-Pro | 19.53 | 20.26 |
| MMLU (aggregate) | 62.22 | 64.80 |
| GSM8K | 39.27 | 45.26 |
| GSM-Symbolic | 18.85 | 22.80 |
| Minerva | 12.52 | 15.63 |
| MATH-500 | 12.52 | 14.48 |
| BigCodeBench | 20.68 | 22.79 |
| HumanEval | 27.10 | 31.38 |
| DS-1000 | 18.58 | 20.40 |
| MBPP | 34.53 | 35.91 |
| MultiPL-E HumanEval | 21.34 | 22.27 |
| MultiPL-E MBPP | 28.67 | 33.96 |
| ARC-E | 90.95 | 92.30 |
| ARC-C | 77.99 | 81.57 |
| PIQA | 72.25 | 80.85 |
| CommonsenseQA | 72.65 | 74.04 |
| SocialIQA | 65.35 | 69.24 |
| HellaSwag | 66.05 | 66.05 |
| WinoGrande | 49.17 | 51.22 |
| LAMBADA-Standard | 66.60 | 66.89 |
| LAMBADA-OpenAI | 72.13 | 72.21 |
| MedMCQA | 32.98 | 33.21 |
| MedQA | 38.81 | 38.96 |

Continued on next page

Table 20 (continued)

| Benchmark | OLMo-3-7B | NCP-8.9B |
|---|---|---|
| Full-suite average (26 tasks) | 46.59 | 49.04 |
| **BPB (↓)** | | |
| HumanEval Gold | 0.384 | 0.364 |
| MBPP Gold | 0.475 | 0.473 |
| MT-MBPP Gold | 0.437 | 0.434 |
| CoQA | 0.366 | 0.370 |
| DROP | 4.474 | 4.397 |
| GSM8K Gold | 0.405 | 0.400 |
| Jeopardy | 0.352 | 0.347 |
| LAMBADA-OpenAI | 0.295 | 0.288 |
| Natural Questions | 0.917 | 0.884 |
| SQuAD | 0.140 | 0.151 |
| BPB average (reported) | 0.824 | 0.811 |

### F.4 Generative evaluation protocol

| Benchmark | Shots | Samples/problem | Metric |
|---|---|---|---|
| GSM8K | 8 | 1 | Pass@1 |
| GSM-Symbolic | 8 | 8 | Pass@1 |
| Minerva Math | 4 | 1 | Exact match |
| MATH-500 | 4 | 32 | Pass@1 |
| BigCodeBench | 3 | 5 | Execution pass@1 |
| HumanEval | 3 | 32 | Execution pass@1 |
| DS-1000 | 3 | 5 | Execution pass@1 |
| MBPP | 3 | 32 | Execution pass@1 |
| MultiPL-E HumanEval | 0 | 32 | Execution pass@1 |
| MultiPL-E MBPP | 0 | 32 | Execution pass@1 |

Table 21: Prompting and sampling settings for the retained generative results. Samples are generated completions per problem, not the inference batch size or the $k$ in pass@$k$.

MMLU uses five-shot multiple-choice evaluation; MMLU-Pro also uses five shots. ARC, CommonsenseQA, PIQA, and SocialIQA use five-shot multiple-choice prompts. HellaSwag uses a five-shot ranked-choice formulation with character-normalized likelihood; WinoGrande uses raw ranked-choice accuracy. MedMCQA and MedQA use token-normalized likelihood. Lambada is evaluated zero-shot using greedy next-token completion. These conventions follow the original large-scale evaluation protocol.

### F.5 Stage-1 checkpoint trajectory

The reported downstream average reaches 46.84 at 700K steps, compared with 46.59 for the final OLMo-3-7B Stage-1 reference.

| Step | 100K | 200K | 300K | 400K | 500K | 600K | 700K |
|---|---|---|---|---|---|---|---|
| Average | 39.64 | 42.38 | 44.27 | 44.78 | 45.66 | 46.06 | 46.84 |
| Step | 800K | 900K | 1.0M | 1.1M | 1.2M | 1.3M | Final |
| Average | 47.52 | 47.85 | 48.05 | 48.00 | 48.64 | 48.87 | 49.04 |

Table 22: Downstream trajectory of the NCP Stage-1 checkpoints. Each average uses the same 26-task convention as the final checkpoint.

## G Related Work

A growing body of research explores high-level prediction in AI, shifting learning objectives from fine-grained, low-level targets to abstract semantic representations. As a pioneering framework, JEPA (Assran et al., 2023; Bardes et al., 2024) emphasizes prediction within a joint-embedding space to bypass the modeling of redundant pixel-level details, a paradigm successfully extended to video understanding (Assran et al., 2025). In the realm of decision-making and planning, world models such as Dreamer (Hafner et al., 2024) and GAIA-1 (Hu et al., 2023) demonstrate that predicting future latent states, rather than raw inputs, is crucial for robust long-horizon reasoning. Similarly, the field of audio has transitioned from raw waveform reconstruction toward the prediction of discrete hidden units (Baevski et al., 2020; Hsu et al., 2021) to better capture long-range dependencies. For multimodal learning, the Large Concept Model (LCM) (team et al., 2024) operates in a modality-agnostic latent space, performing inference on sentence-level embeddings rather than tokens. Collectively, these developments suggest that higher-level prediction is a universal path toward building better models.

Several works have explored language modeling beyond individual tokens. One prominent approach involves structural compression, where tokens are aggregated into patches: MegaByte (Yu et al., 2023) uses a fixed chunk size to convert bytes into patches; BLT (Pagnoni et al., 2024) and H-net (Hwang et al., 2025) dynamically patchify byte sequences into patches, and DLCM (Qu et al., 2026) extends the H-net paradigm to the token level. They then perform patch-level language modeling. High-level processing methods also focus on accelerating model training or inference (Shao et al., 2025; Liu & Zhang, 2025). More recently, latent reasoning paradigms (Su et al., 2025; Hao et al., 2025; Tack et al., 2025; Deng et al., 2024) have emerged to leverage latent tokens for capturing abstract representations. Despite these advancements, designing a self-supervised pretraining objective specifically for high-level language representations remains an open challenge.