

Breaking the Token Ceiling: Distilling Smaller, Stronger Byte Models

Kalyani Marathe^{1†*}, Artidoro Pagnoni^{2†*}, Tomasz Limisiewicz¹², Margaret Li¹,
Mike Lewis^{2#}, Luke Zettlemoyer^{12#}, Srinivasan Iyer^{2†#*}

¹University of Washington, Seattle

[‡]Work done at Meta FAIR

²Meta FAIR

[†]Joint first author. [#]Joint last author.

Abstract

Small models are made more capable through distillation from a larger one that shares their tokenization scheme. However, do distilled byte and token models behave similarly in terms of scaling trends as compute and data increases? To enable this comparison, we introduce two variants to efficiently convert token logits to **Byte Logits**: 1) approximate: **Marginalize-It**, and 2) exact: **End-Of-Token**. We then present the first large scale study of overtraining decoder-only dense transformer models varying two dimensions simultaneously: the **tokenization scheme** (**Tokens**, **Bytes**, **Bytes w/ <eot>**) and the **training objective** (Distillation vs. Cross-Entropy), sweeping layer-parameter-matched models with ≈ 1 billion parameters up to 1 trillion bytes of data. Across eight benchmarks spanning three categories: Multiple Choice QA, Language Generation, and Machine Translation, we find that **Token-1B** models outperform byte models (**End-Of-Token-1B** and **Bytes-1B**) in the low-FLOP regime but eventually plateau; byte models start worse yet surpass **Token-1B** models with more compute, reaching a higher downstream task performance ceiling. Extrapolating the average top-1 error vs. validation BPB scaling laws predicts that, asymptotically, distilled **End-Of-Token-1B** outperforms distilled **Token-1B** by up to 4% on averaged downstream task performance. They are also far more data efficient, matching the performance of distilled **Token-1B** using only **one-sixth** of the training data. Moreover, by operating over a small vocabulary of ≈ 256 bytes instead of $\approx 100K$ tokens, they circumvent the need for top- k truncation during logit dumping, while also reducing logit storage costs to roughly **one-fifth**. Finally, our downstream performance scaling laws predict that our distilled **End-Of-Token-1B** models asymptotically surpass the Llama 3.2-1B model (Meta AI, 2024), Gemma-3-1B-pt (Kamath et al., 2025), and Gemma 2B (Team et al., 2024) models on averaged downstream tasks by up to 6.5%, 8.1%, and 2.1%, respectively.

1 Introduction

Scaling has become the prevailing approach for improving language model performance (Henighan et al., 2020; Hoffmann et al., 2022; Kaplan et al., 2020). However, deploying models in real-world settings demands more than scale alone: the cost of serving can outweigh the benefits of scale (Dettmers & Zettlemoyer, 2023; Pope et al., 2023). Models are often 1) distilled into smaller yet capable models that, by training against the teacher’s next-token distribution rather than ground-truth tokens, outperform on downstream tasks (Beyer et al., 2022; Hinton et al., 2015), and 2) overtrained to amortize inference costs over a model’s lifetime (Gadre et al., 2024; Sardana et al., 2023). Yet it remains unclear how the choice of student tokenization interacts with these techniques as compute scales.

In this work we compare how bytes and tokens affect distillation scaling trends as the compute budget grows. Bytes are a compelling alternative for two reasons: First, by compressing the vocabulary to just ≈ 256 values, transformers operating on bytes can preserve the distribution of a large token language model exactly (Hayase et al., 2025; Phan et al., 2024) and potentially can bypass the need for top- k logit truncation for offline distillation. Second, they also possess interesting properties: they learn relationships between bytes from fewer samples than token models require for tokens (Hestness et al., 2017),

*Correspondence: kmarathe@uw.edu, {artidoro, sviyer}@meta.com

which makes them data efficient learners (Xue et al., 2022; Zheng et al., 2025). Distillation into such models, though, hinges on open questions: 1) how to efficiently convert token logits to byte logits, 2) how does tokenization and training objectives affect the BPB vs FLOPs power laws 3) whether distillation outperforms supervised byte (and token) level training on downstream tasks, 4) at what compute budget byte-level surpasses token-level distillation, 5) how the trained models and the asymptotic predictions compare to the open-weights model of the comparable model sizes and what are the data and storage efficiency implications. This work addresses all five. We summarize our key contributions in section 1.1 below:

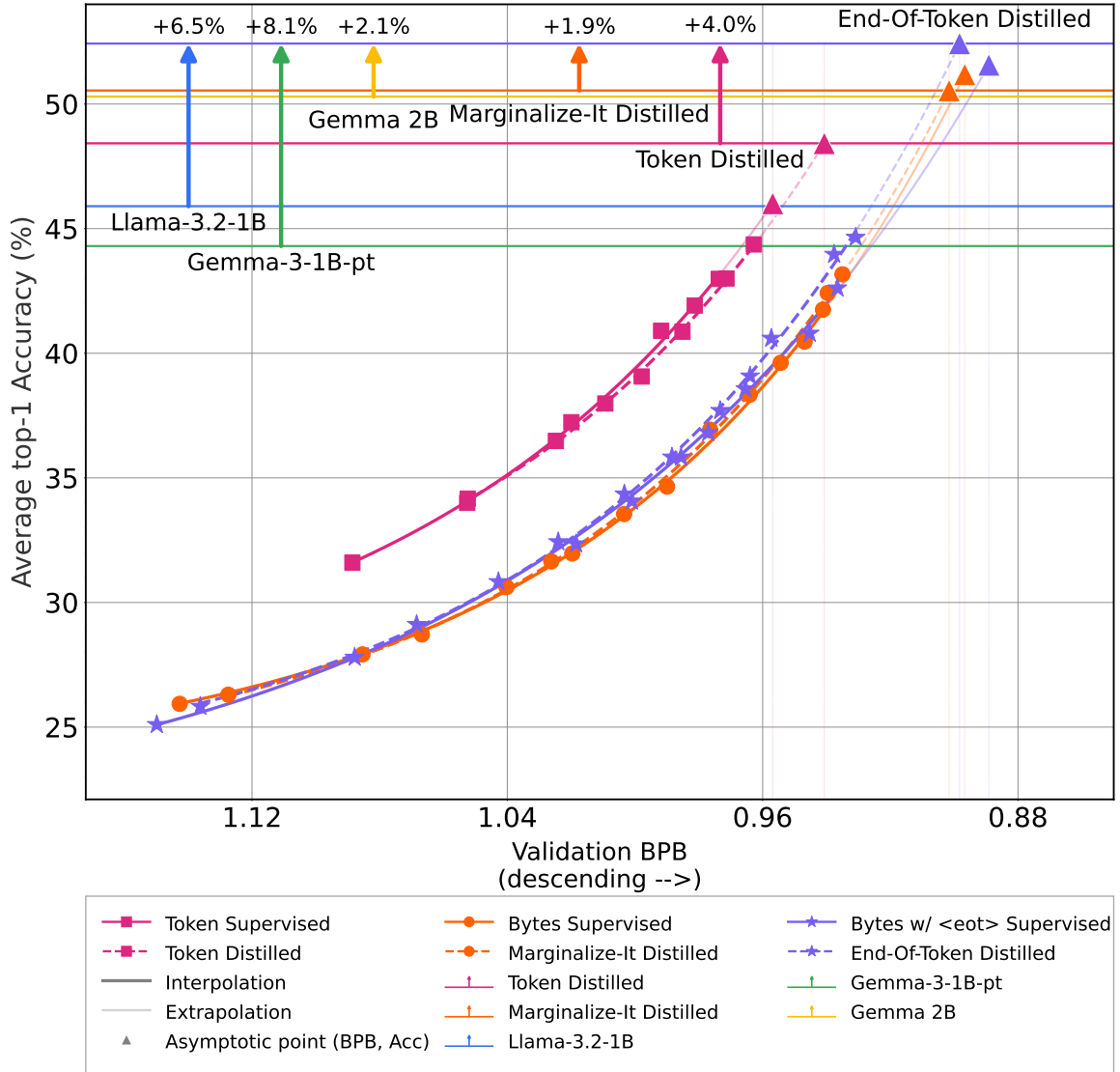


Figure 1: Downstream task performance scaling laws for $lr=4e-3$ averaged across six benchmarks: Multiple Choice QA (ARC-Easy, ARC-Challenge, HellaSwag, PIQA) and Language Generation (MBPP, Natural Questions) tasks. **End-Of-Token Distilled** model, by preserving the teacher distribution exactly as well as with the highest amount of compute spent on fixed amount of data raises the asymptotic performance ceiling. **End-Of-Token Distilled** asymptotically outperforms the **Token Distilled** model by 4%, the **Marginalize-It Distilled** model by 1.9%, and the three open weights models: Llama 3.2-1B model (Meta AI, 2024), Gemma-3-1B-pt (Kamath et al., 2025), and Gemma 2B (Team et al., 2024) models on averaged downstream tasks by upto 6.5%, 8.1%, 2.1% respectively.

1.1 Summary of Contributions

Token logits to Byte logits conversion in a single forward pass. Distilling a smaller byte model from a large token model first requires converting the teacher’s token logits onto byte logits. A token comprises several bytes, so the probability distribution must be split across byte positions. Prior methods (Hayase et al., 2025; Phan et al., 2024) to convert token distribution to byte distribution require multiple forward passes which is prohibitive when teacher inference is computationally expensive. We introduce two single-pass variants for converting token logits to **Byte Logits** by marginalizing over byte positions: 1) approximate: **Marginalize-It** re-normalizes over the prefix-matching tokens, while 2) exact: **End-Of-Token** adds an `<eot>` token to the vocabulary so no probability is lost.

Bits-Per-Byte vs Training FLOPs Power Laws Across Tokenization and Training Objectives. With a logit conversion in hand, we ask how the resulting byte models scale compared to token models and their supervised counterparts. We present a large-scale study of overtraining layer-parameter matched decoder-only dense transformers along two axes: the training objective (Distillation loss vs Supervised/Cross-Entropy loss) and the tokenization scheme (**Tokens** , **Bytes** , **Bytes w/ <eot>**): sweeping decoder-only transformer models (Vaswani et al., 2017) with fixed 1.28 billion layer parameters up to 1 trillion bytes data. We train multiple **Token-1B** (1.81B), **Bytes-1B** (1.28B), and **End-Of-Token-1B** (1.28B) models to fit Validation BPB vs Training FLOPs power laws to show that distilled **End-Of-Token-1B** models show better asymptotic behavior compared to distilled **Token-1B** and **Bytes-1B** .

Average Downstream Task Accuracy vs Training FLOPs Scaling Trends. To understand how the scaling trends in validation loss translate into downstream task performance for **Marginalize-It** (approximate) and **End-Of-Token** (exact) we now turn to benchmark evaluations. Across three categories and eight benchmarks: Multiple Choice QA (ARC-Easy, ARC-Challenge, HellaSwag, PIQA), Language Generation (MBPP, Natural Questions), and Machine Translation (Flores) we find that **Token-1B** transformers (supervised and distilled) with show strong performance at low compute regimes but plateau quickly. **Bytes-1B** (1.28B) and **End-Of-Token-1B** (1.28B), lag initially behind the **Token-1B** (1.81B) models, continue to improve at a steeper rate with additional compute despite fewer total parameters due to the small vocabulary size of ≈ 256 bytes.

Comparing Distillation Scaling Trajectories Across Compute Budgets. Downstream task performance vs training FLOPs scaling trend raises the practical question of which method to choose at a given compute budget. We fit Downstream error vs. Validation BPB scaling laws (Figure 1) and introduce **Feather Plots** (Figure 11), which connect IsoFLOP points on the downstream-error vs. Validation-BPB curves. These plots let us compare downstream performance trajectories across compute budgets for each of the tokenization scheme. By extrapolating the downstream error vs validation BPB scaling laws, our experiments predict that distilled **End-Of-Token-1B** transformers raise the asymptotic ceiling of the averaged downstream task performance. We then predict that by both preserving the teacher distribution exactly as well as by spending $\approx 30.94\%$ more compute on a fixed amount of data compared to **Marginalize-It** (approximate) method, distilled **End-Of-Token-1B** (exact) improves over distilled **Token-1B** by up to **4%** and distilled **Bytes-1B** by up to **1.9%** asymptotically.

End-Of-Token-1B : Data and Storage Efficiency, Asymptotic win over Open-Weight models. The asymptotic accuracy gains from **End-Of-Token** method also come with efficiency gains. Distilled **End-Of-Token-1B** model also surpass distilled **Token-1B** models with up to **one-sixth** of the training data with sufficient training despite having fewer total parameters. By operating over a small vocabulary of ≈ 256 bytes instead of the $\approx 100K$ tokens they circumvent the need for top-k truncation during logit dumping, reducing logit storage costs to roughly **one-fifth**. Finally, comparisons with the open-weight models with the asymptotic predictions reveal our distilled **End-Of-Token-1B** model surpass the **Meta AI** (2024), **Kamath et al.** (2025), and **Team et al.** (2024) models on averaged downstream tasks by upto **6.5%** and **8.1%**, **2.1%** respectively.

Paper Outline: We introduce two logit conversion methods: **Marginalize-It** and **End-Of-Token** to convert the token logits to byte logits (Section 2). We then describe the experimental setup, the six scenarios used for our scaling study, and the benchmarks used for evaluations (Section 3). We then study the validation BPB vs FLOPs scaling laws (Section 4), the downstream task performance vs training FLOP scaling trends (Section 5) and the downstream performance vs validation BPB scaling laws (Section

6). We then use these scaling laws to find out when byte distillation will be advantageous over token distillation baseline (Section 7). Through the lens of downstream error scaling laws, with the help of **Feather Plots** we demonstrate how the relative performance ordering between the **Token Distilled**, **Marginalize-It Distilled**, and **End-Of-Token Distilled** models evolve with the training compute budget. We then benchmark our trained checkpoints and our asymptotic predictions against the open weight models (Section 8). Next, we contextualize our work within the existing literature (Section 9). Lastly we reflect on the findings, limitations and opportunities for future work (Section 10).

2 Logit Conversion Methods

In this section we describe two methods: **Marginalize-It** and **End-Of-Token**, we use in our scaling study to efficiently convert token logits to **Byte Logits**. Figure 2 demonstrate the byte logits formation process with the help of an example. We provide the pseudocode for the two methods in Appendix F.1 and discuss special cases in Appendix F.2.

2.1 Marginalize-It Logits

For each token used in teacher inference, we have a $|V|$ -sized logit tensor denoting the next-token distribution, where V is the vocabulary of the BPE tokenizer. We also have the decoded byte sequence of every BPE token in V . To compute the *first* byte distribution of each BPE token, we marginalize over the full vocabulary V based on the first byte. For each *subsequent* byte, we restrict the vocabulary to the tokens whose prefixes match the ground-truth bytes and compute the byte distribution conditioned on this subset. We refer to this as the **Marginalize-It** method. While this approach always yields an exact distribution for the first byte, the distributions over subsequent bytes are only *approximate*. Figure 2 (column 1) shows the **Marginalize-It** logit conversion process step by step. To predict the distribution B3, the continuations of token **is** (whose probability is 0.125) are silently dropped and only the tokens **isu** and **isk** are carried forward. While obtaining the next token distribution of token **is** can fix this problem, it requires running additional teacher inference over the sequence: $[\langle \text{bos} \rangle, \text{T}, \text{iram}, \text{is}]$. The number of additional inferences required to obtain exact conversions can get intractable for long sequences. To avoid that problem, in **Marginalize-It** method remaining probability is redistributed among tokens **isu** and **isk**:

$$P(\text{isu}) = \frac{0.5}{0.5 + 0.125} = 0.8, \quad P(\text{isk}) = \frac{0.125}{0.5 + 0.125} = 0.2,$$

During training, the byte sequence used is: $[\langle \text{bos} \rangle, \text{T}, \text{i}, \text{r}, \text{a}, \text{m}, \text{i}, \text{s}, \text{u}, \langle \text{eos} \rangle]$. We provide additional details in Appendix F.2. **Marginalize-It** serves as the main baseline throughout the study.

2.2 End-Of-Token Logits

The **End-Of-Token** method is essentially the **Marginalize-It** method with two key changes. First, while forming the byte logits after teacher inference, we append an $\langle \text{eot} \rangle$ token to mark the end of each BPE token as shown in Figure 2 (column 2). Second, we expand the byte vocabulary by one to store the $\langle \text{eot} \rangle$ probabilities and then apply marginalization over byte positions to form the byte logits. During training, we append $\langle \text{eot} \rangle$ after every BPE token in the pretraining dataset: $[\langle \text{bos} \rangle, \langle \text{eot} \rangle, \text{T}, \langle \text{eot} \rangle, \text{i}, \text{r}, \text{a}, \text{m}, \langle \text{eot} \rangle, \text{i}, \text{s}, \text{u}, \langle \text{eot} \rangle, \langle \text{eos} \rangle, \langle \text{eot} \rangle]$. This resolves the missing-probability problem of **Marginalize-It**. To compute B3 distribution, unlike **Marginalize-It**, **End-Of-Token** considers *all* tokens with prefix **is**. Since the total probability of tokens starting with prefix **is** is $0.5 + 0.125 + 0.125 = 0.75$:

$$P(\text{isu}\langle \text{eot} \rangle) = \frac{0.5}{0.75} = 0.667, \quad P(\text{isk}\langle \text{eot} \rangle) = \frac{0.125}{0.75} = 0.167, \quad P(\text{is}\langle \text{eot} \rangle) = \frac{0.125}{0.75} = 0.167.$$

In this way, the probability mass of the possible continuations of **is** is absorbed into the $\langle \text{eot} \rangle$ token. This approach is exact, does not require additional inferences over teacher. Inserting $\langle \text{eot} \rangle$ tokens also has an unintended but remarkable consequence: the $\approx 30.94\%$ additional compute it spends per unit of data (one $\langle \text{eot} \rangle$ every approx 4.5 bytes) turns out to further improve asymptotic performance relative to **Marginalize-It** method.

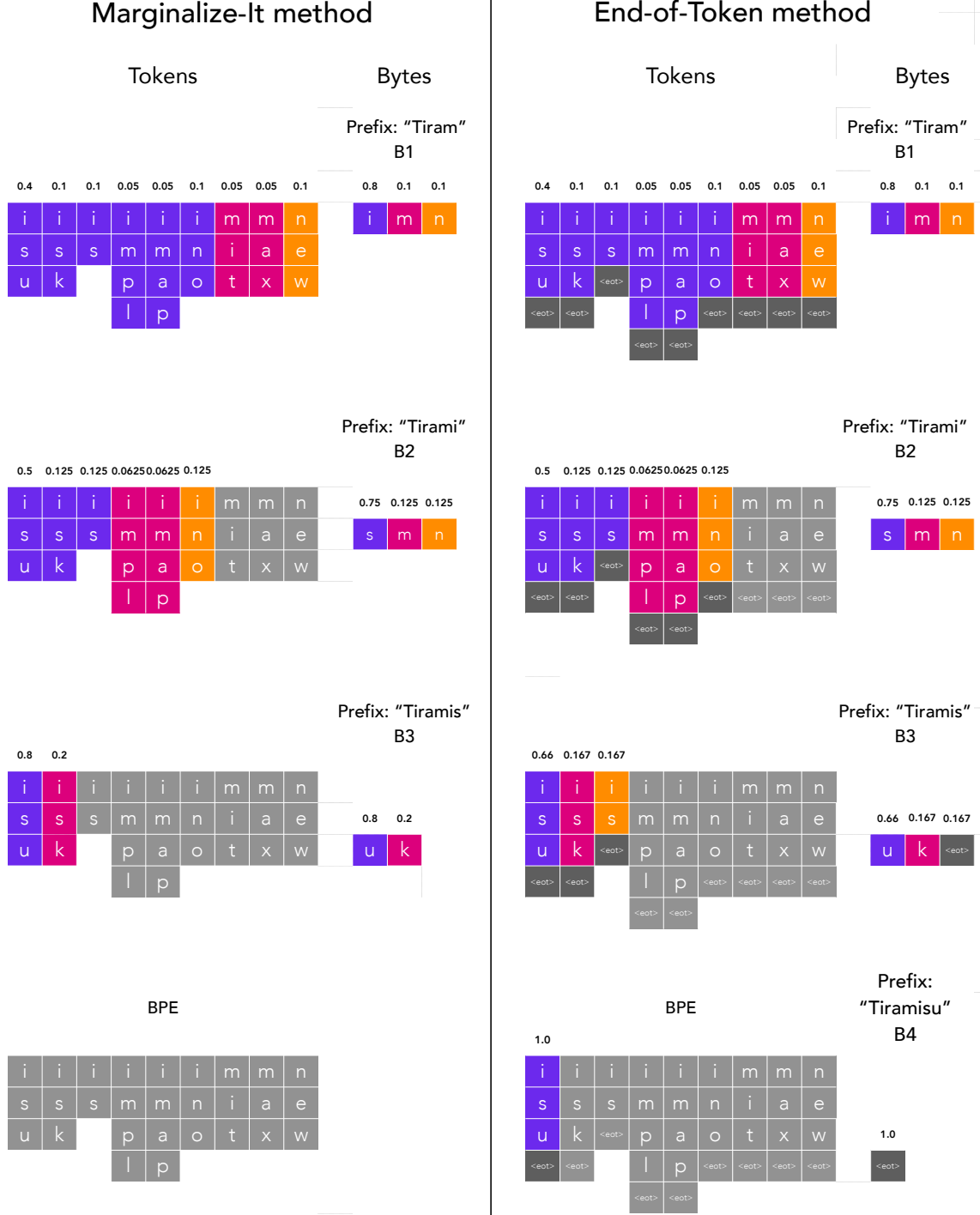


Figure 2: A step-by-step illustration of the two logit conversion methods. The first column demonstrates the **Marginalize-It** Byte Logits formation process. Consider the tokenized prompt `Tiramisu` = `[<bos>, T, iram, isu, <eos>]`, and in particular the next-token distribution following the token `iram`, for which the ground truth is `isu`. Each column represents the decoded byte sequence of a BPE token from the full vocabulary (pruned here for clarity) of the Llama 3-8B model, with the token’s probability shown at the top of the column; columns sharing a color share a byte prefix. To compute the first-byte distribution, the columns sharing the byte prefix **B1** are aggregated. Next, we subset the vocabulary by the ground-truth byte `i` and aggregate the probabilities over **B2**. Finally, we repeat this process for **B3**. A limitation of the **Marginalize-It** method is that it does not account for all possible continuations after the BPE token `is` while predicting **B3**: here, for example, only the tokens `isu` and `isk` are considered. The **End-Of-Token** method (second column) addresses this problem by preserving the language modeling distribution exactly, assigning the residual probability mass to the `<eot>` token, as illustrated for the prediction of byte **B3**.

3 Scaling Study Design

3.1 Experimental Setup

In this section we describe the Model Architecture, Pretraining Dataset and Training Details. We provide specifics of these settings in Section A.

Model Architecture. We use the transformer architecture (Vaswani et al., 2017) to train all of the layer-parameter-matched models with ≈ 1 billion parameters. We use three variants: **Token-1B**, **Bytes-1B**, and **End-Of-Token-1B** listed in Appendix (Table 9). While the three model architectures have equal total layer parameters (1.28B), the **Token-1B** (1.81B) is in fact larger due to the larger vocabulary size. We stick to the Llama-3 (Grattafiori et al., 2024) implementation for the model design choices.

Training Dataset. We use the Llama-2 (Meta AI, 2024) training mixture for our supervised and distillation experiments. Our study is designed such that both the token logits and byte logits occupy the same amount of memory for the same amount of data. See section A.6 for the full storage cost breakdown.

Training Details. We train our models with the AdamW (Loshchilov & Hutter, 2017) optimizer and sweep three learning rates $\in \{1e-3, 4e-3, 8e-3\}$. We use 10% warmup steps for each data scale. We provide additional details on the loss function in Appendix (A.3)

3.2 Scaling Along Two Axes: Tokenization and Training Objectives

Our study seeks to explore scaling trends of layer-parameter-matched models with overtraining for six experiments described below. This enables us to understand the complex relationships between the Supervised (Cross-Entropy Loss) and Distillation pretraining as well as the effects of the tokenization schemes: **Tokens**, **Bytes**, **Bytes w/ <eot>** on the training dynamics. Table 1 details the six experimental settings in our overtraining study. Model size and FLOPs/unit values are obtained from Table 9. For **Marginalize-It** and **End-Of-Token** distillation we use the logit conversion process described in Section 2.

Experiment	Tokenization	Vocab. size	Objective	Architecture	Model size	FLOPs/unit
Token Supervised	Tokens (Llama 3-8B)	128,256	Cross-Entropy	Token-1B	1.81B	8.18×10^9
Token Distilled	Tokens (Llama 3-8B)	128,256	Distillation (top-k)	Token-1B	1.81B	8.18×10^9
Bytes Supervised	Bytes (256 + 4 special)	260	Cross-Entropy	Bytes-1B	1.28B	8.81×10^9
Marginalize-It Distilled	Bytes (256 + 4 special)	260	Distillation (approximate)	Bytes-1B	1.28B	8.81×10^9
Bytes w/ <eot> Supervised	Bytes w/ <eot> (256 + 5 special)	261	Cross-Entropy	End-Of-Token-1B	1.28B	9.44×10^9
End-Of-Token Distilled	Bytes w/ <eot> (256 + 5 special)	261	Distillation (Exact)	End-Of-Token-1B	1.28B	9.44×10^9

Table 1: Six experiments in our overtraining study across the training-objective and tokenization axes.

3.3 Benchmark Evaluations

Multiple Choice Question Answering. We evaluate performance on four benchmarks which evaluate the ability of a model to solve multiple choice questions. Specifically we use ARC-Easy (P. Clark et al., 2018), ARC-Challenge (P. Clark et al., 2018), HellaSwag (Zellers et al., 2019), and PIQA (Bisk et al., 2020) to benchmark the token and byte checkpoints with and without distillation.

Language Generation Tasks. We further evaluate the language modeling capabilities of the byte and token models on MBPP (Austin et al., 2021) and Natural Questions (Kwiatkowski et al., 2019).

Machine Translation. Lastly we also evaluate our trained checkpoints on English to German and German to English translation using the Flores (Goyal et al., 2022) benchmark.

4 Scaling Trends I: Validation BPB vs Training FLOPS

In this experiment we analyze the overtraining behavior of the six experiments described in 3.2. We plot the BPB of the model on a held out validation dataset.

4.1 Validation BPB vs Training FLOP Plots

We fit the power law for validation BPB with respect to FLOPs using equation (1) where x is the number of FLOPs and y is the BPB at a given compute budget (Table 2). The y , asymptotically approaches c :

$$y = b \cdot x^a + c \quad (1)$$

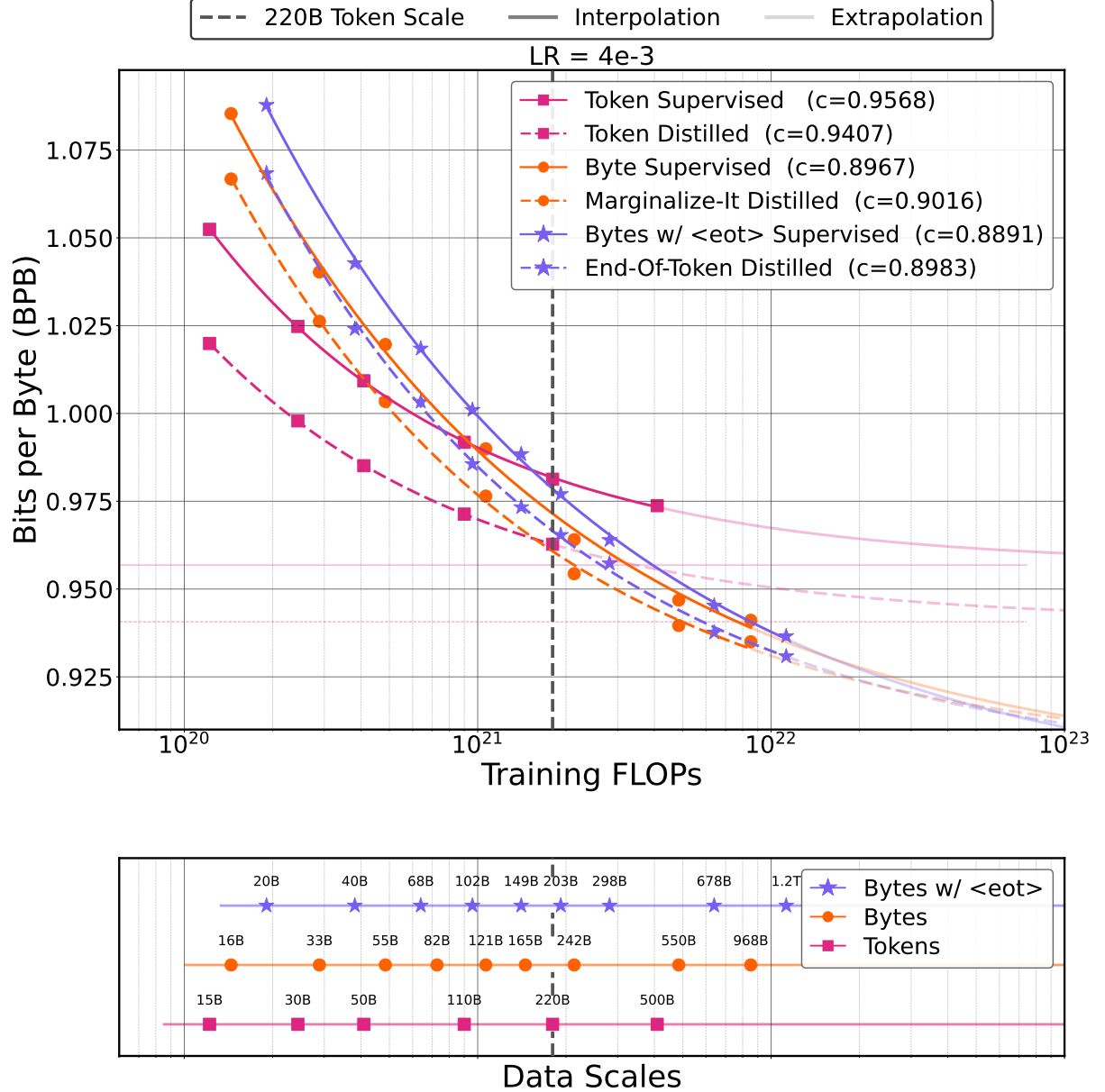


Figure 3: The figure shows our scaling curves for different data scales, for a learning rate of $4e-3$ and a fixed parameter count of approximately 1 billion parameter transformer models (see Table 9 for details). Note that data scales plot serves as an alternative x axis to training flops. Since the FLOPs per unit changes for the three models, we specify the number of units used for training. The distilled models are trained using Llama 3-8B as a teacher model. The validation BPB of the teacher model is 0.85511.

Table 2: Power-law scaling equations $y = b \cdot x^a + c$ fitted to BPB vs. training FLOPs for LR=4e-3.

Scenario	Equation	R^2
Token Supervised	$y = 1.075 \times 10^9 \cdot x^{-0.500369} + 0.9568$	0.9999
Token Distilled	$y = 2.803 \times 10^8 \cdot x^{-0.475375} + 0.9407$	1.0000
Bytes Supervised	$y = 4.467 \times 10^6 \cdot x^{-0.365844} + 0.8967$	0.9980
Marginalize-It Distilled	$y = 2.706 \times 10^7 \cdot x^{-0.407427} + 0.9016$	0.9974
Bytes w/ <eot> Supervised	$y = 3.080 \times 10^6 \cdot x^{-0.354594} + 0.8891$	0.9982
End-Of-Token Distilled	$y = 2.734 \times 10^7 \cdot x^{-0.404740} + 0.8983$	0.9990

Table 3: We observe that the amount of FLOPs spent on a fixed amount of data (one BPE token in this case) affects the asymptotic BPB. Both supervised and distilled models asymptotically follow descending BPB trend as the Total FLOPs Per Token increase. The FLOPs/Unit values are obtained from Table 9. Depending on the model architecture, a unit can be a token or a byte. We note that **End-Of-Token-1B** uses $\approx 30.94\%$ extra FLOPs per token compared to **Bytes-1B**.

Method	Architecture	FLOPs / Unit	Units Per Token	Total FLOPs Per Token	Asymptotic BPB
					4e-3
Token Supervised	Token-1B	8.18×10^9	1.0	8.18×10^9	0.9568
Bytes Supervised	Bytes-1B	8.81×10^9	4.5	39.65×10^9	0.8967
Bytes w/ <eot> Supervised	End-Of-Token-1B	9.44×10^9	5.5	51.92×10^9	0.8891
Token Distilled	Token-1B	8.18×10^9	1.0	8.18×10^9	0.9407
Marginalize-It Distilled	Bytes-1B	8.81×10^9	4.5	39.65×10^9	0.9016
End-Of-Token Distilled	End-Of-Token-1B	9.44×10^9	5.5	51.92×10^9	0.8983

We notice several trends from these plots:

1. First, both **Marginalize-It Distilled** (approximate) and **End-Of-Token Distilled** (exact) methods demonstrate improvements over their supervised counterparts.
2. **Marginalize-It Distilled** training, despite being an approximation works very well in practice. **End-Of-Token Distilled** method is exact and yields consistent improvements over the supervised training.
3. Second, we also observe that **Token-1B** models have the lowest BPB at lower compute budgets, and appear to saturate faster. The **End-Of-Token-1B** have the worst BPB to begin with yet yield the best asymptote in the infinite compute regime. **Bytes-1B** training runs, appear to be sandwiched between the other two scenarios at the beginning and the end of the training runs.
4. From Table 3, we observe that asymptotically for both supervised and distilled models: 1) **End-Of-Token-1B** BPB < **Bytes-1B** BPB < **Token-1B** BPB. We also observe that the supervised and distilled models with the same tokenization scheme exhibits small differences in the asymptotic value.
5. We also observe interesting trends across all training runs. At around 1.8×10^{21} FLOPs, the **Token Supervised** training run appears to have the highest BPB of all, with all other five curves demonstrating lower BPB values. Does this mean that all other five models at that compute budget are better than **Token Supervised**? To find out, we turn to benchmark evaluations in Section 5.

5 Scaling Trends II: Task Performance vs Training FLOPs

In this experiment we seek to understand if the observations from the validation BPB curves from section 4 are consistent on a variety of benchmarks. We evaluate the capabilities of our trained models on a total of eight benchmarks spanning three categories: Multiple choice question answering, Generative, and Machine Translation tasks (Refer to section 3.3 for more details).

5.1 Results on Multiple Choice Question Answering Benchmarks

To evaluate the capabilities of trained models on four multiple choice question answering tasks we plot the % Accuracy vs Training FLOP trends for our six experiments in Figure 4. In addition we also report the averaged accuracy across four benchmarks. We observe that:

1. The averaged choice accuracy improves for all of the 6 experiments, and the distilled models consistently show better performance compared to the supervised models. This trend also holds for the benchmarks individually.
2. **Token-1B** models start best, outperforming all of the byte curves at lower compute budgets. All of the byte curves start worse but appear to be improving faster than the token models, eventually catching up with them.
3. If we compare within the four byte experiments, both supervised and distilled **End-Of-Token-1B** start worse, and eventually surpass the **Bytes-1B** models methods respectively.
4. **Token-1B** models, both supervised and distilled have the best performance at the beginning and appear to be eventually saturating, a pattern also observed in Validation BPB vs Training FLOPs curves.
5. Lastly we observe that at 220B token supervised budget only the distilled **Token-1B** outperforms its supervised counterpart curve but the byte curves, both supervised and distilled **Bytes-1B** and **End-Of-Token-1B** models are much worse, which is counterintuitive to the key observation in 5.

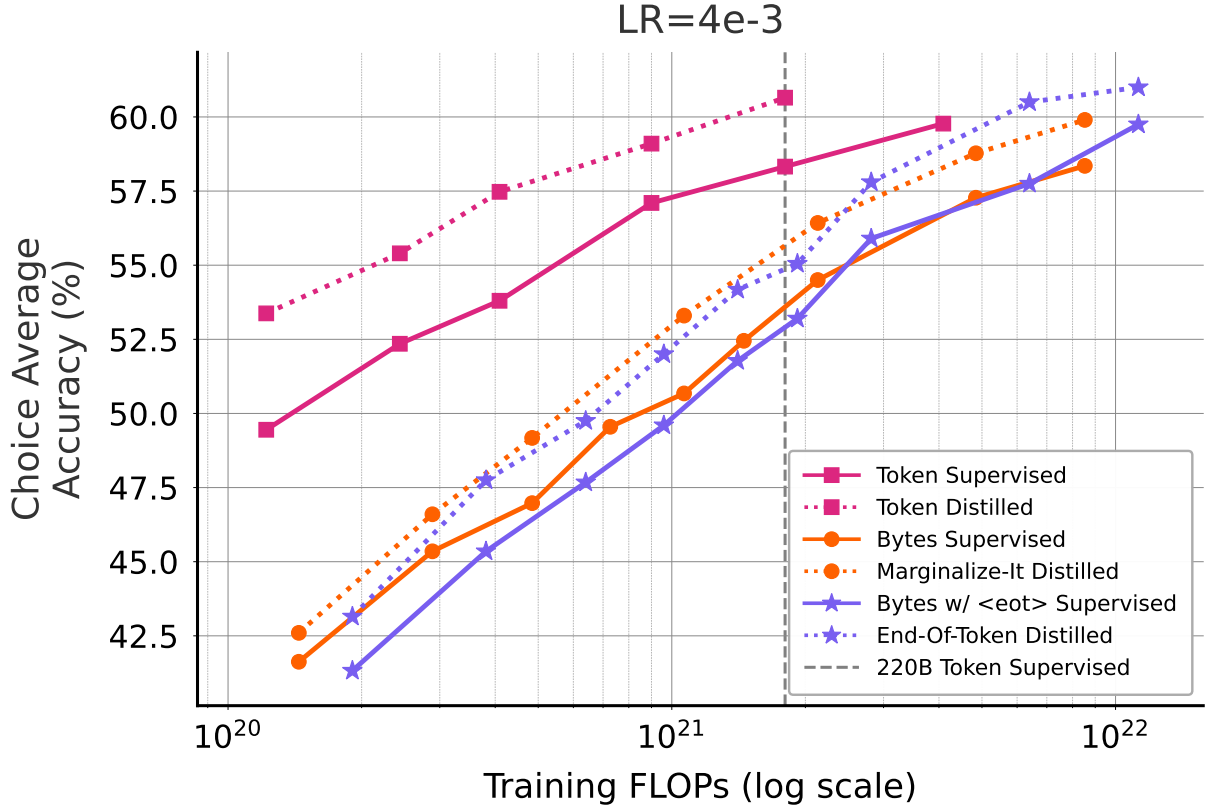


Figure 4: We plot Accuracy vs training FLOP curves for all the six scenarios described in section 3.2. For all of the Multiple Choice Question Answering Benchmarks, we observe that the token models show better performance at low compute budgets and eventually plateau. **Bytes-1B** and **End-Of-Token-1B** models: both supervised and distilled, start worse yet improve at a much faster rate.

5.2 Results on Language Generation Benchmarks

In this experiment we study the language generation capabilities of the trained models on two benchmarks (Figure 5) (more details in section 3.3). We observe that:

1. The trends are similar to Multiple Choice benchmarks. However the saturation effect of **Token-1B** models seems to be less pronounced, potentially because the tasks are much harder.
2. In all cases, distilled models appear to be better for the same compute budget and the result about observation 5 at 1.8×10^{21} FLOPs budget is observed in this case as well. Only the **Token Distilled** model has better performance while all other four byte curves are much worse in most cases.

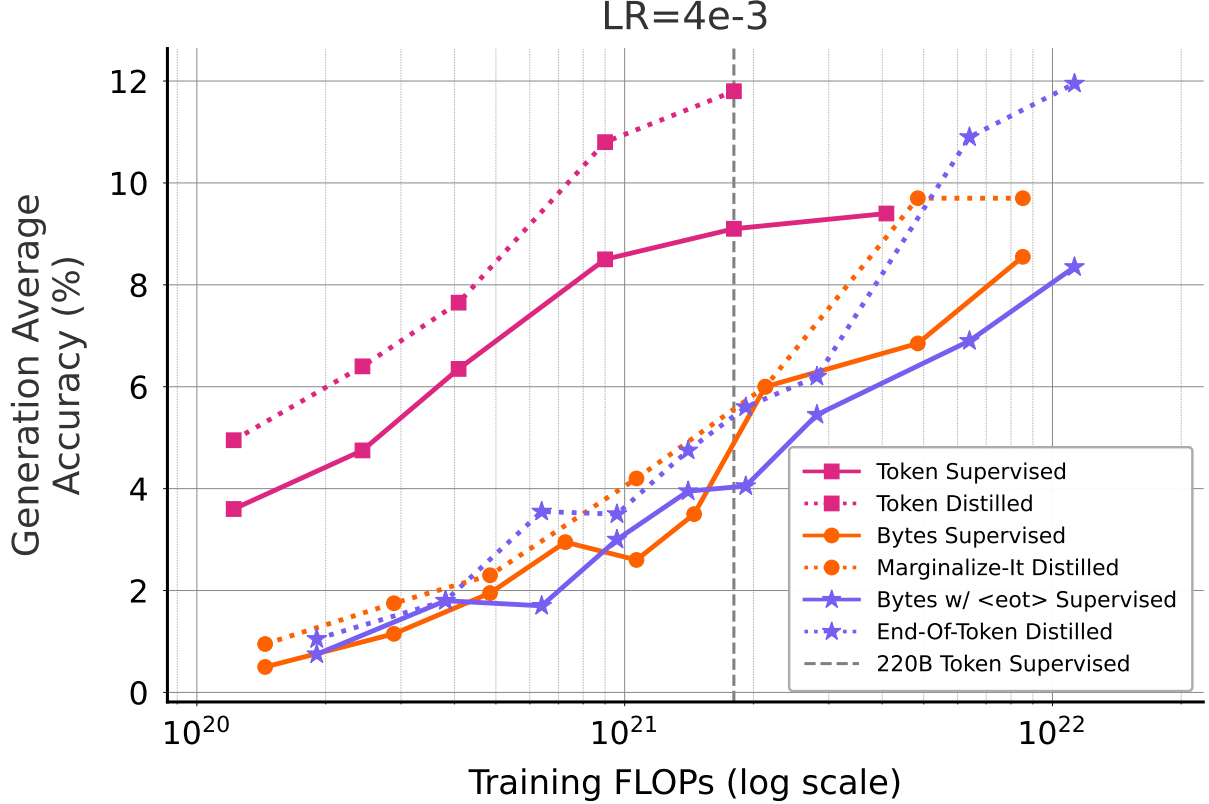


Figure 5: We visualize % Accuracy vs training FLOPs curves for language generation tasks. On generative tasks, token models achieve better performance at lower compute budgets and appear to plateau with the training FLOPs. **Byte** models and **Bytes w/ <eot>** models show a different scaling behavior, with much worse performance at a lower compute budget which continues to improve with FLOPs with no sign of saturation.

5.3 Results on Machine Translation Benchmarks

We also evaluate models on machine translation benchmarks and plot BLEU score (Papineni et al., 2002) vs Training FLOP curves. The plots (Figure 6) show that:

1. Byte model BLEU scores appear to be following an S-shaped trend with training FLOPs, improving slowly at low compute regimes, followed by a steeper improvement regions and then finally saturating at high compute budgets.
2. Next, the results on these benchmarks contradict the observation 5 here as well.
3. **End-Of-Token Distilled** models by preserving the teacher distribution exactly show significant improvements over **Bytes w/ <eot> Supervised** models.

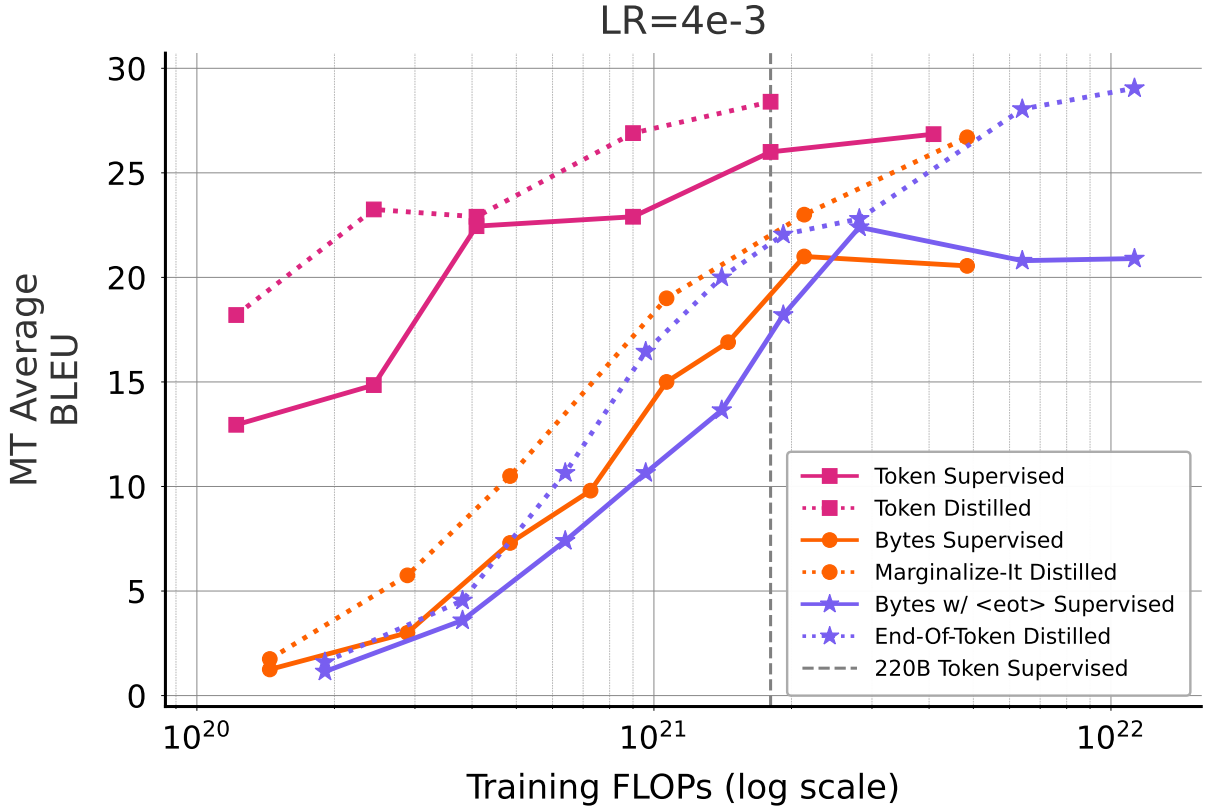


Figure 6: We plot BLEU score for the translation tasks for different training budgets. We observe clearly different scaling behaviors for token and byte models. Token models can achieve significantly better performance at smaller compute budgets. The BLEU score appears to saturate with overtraining. Byte models both with and without the `<eot>` demonstrate extremely low BLEU scores at lower compute budgets yet the rate of improvement with overtraining is better compared to token models.

5.4 Concluding Remarks

In section 5.1, 5.2, and 5.3, we consistently saw that despite achieving better validation BPB at $\approx 1.8 \times 10^{21}$ FLOPs, all four byte models, both distilled and supervised continue to be much worse compared to **Token Supervised** model. What could explain this behavior? In section 6 we set out to explain this phenomenon.

For the data points we observe, the distilled models achieve lower BPB and better overall downstream task performance than their supervised counterparts. However, this does not always hold when comparing the **Token-1B** against the **Bytes-1B** and **End-Of-Token-1B** models: a **Token-1B** can have worse BPB yet better downstream performance, as observed at the 220B token data scale.

6 Scaling Trends III: Task Performance vs Validation BPB

We now turn to explaining how the observations in sections 4 and 5 fit together. For all of the three downstream task categories we study the relationship between the downstream task performance and the validation BPB. To do so, we use the formulation of scaling laws for downstream tasks in (Gadre et al., 2024). We plot the Average top-1 error, essentially, $(1 - \% \text{ Accuracy})$ against the validation BPB, for Multiple Choice Question Answering (section 6.1) and Language Generation (section 6.2). Lastly, since Machine Translation (section 6.3) uses BLEU as an evaluation metric, we plot the BLEU score vs the validation BPB.

6.1 Results on Multiple Choice Question Answering Benchmarks

1. Across all benchmarks as well the averaged performance (Figure 7), all of the six scaling trends demonstrate significantly different behavior. First, if we compare the performances for constant BPB, they imply wildly different error rates. Second, the iso-error lines intersect with the **End-Of-Token** and **Token** curves at very different compute budgets.
2. Second: Changes in the tokenization scheme i.e. the amount of FLOPs spent to model a fixed amount of data can dramatically change the scaling behavior. For instance, **Bytes-1B** and **End-Of-Token-1B** models do not show significantly different scaling behaviors compared to **Token-1B** models.
3. Lastly, the training objective (supervised as well as distillation) also appear to alter the scaling behavior.

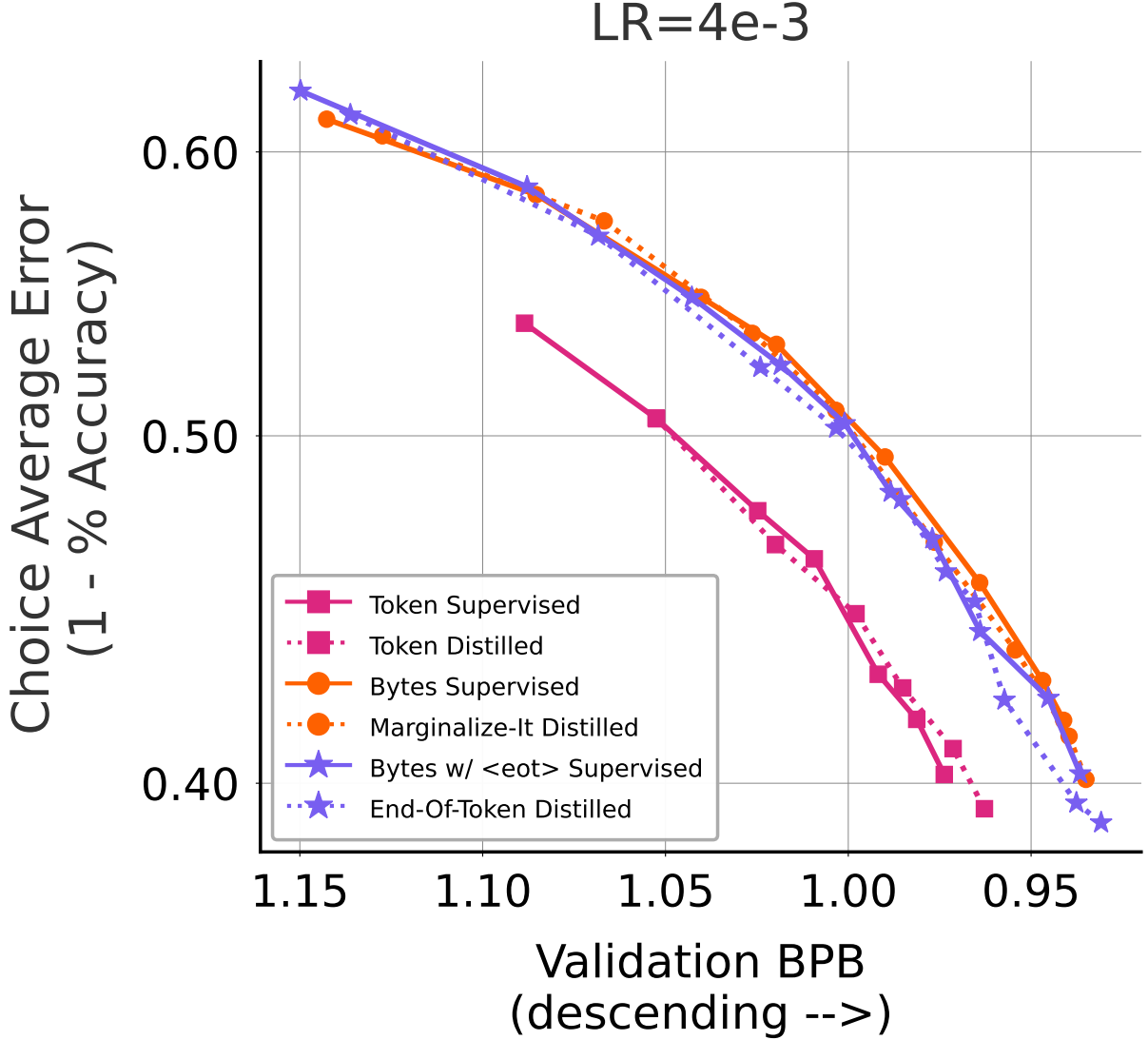


Figure 7: We visualize Average-top-1 error vs the validation BPB curves for the six scenarios for $LR=4e-3$. All of these curves including the averaged metric appear to follow an exponential decay. **Token-1B** shows significantly different trend compared to **Bytes-1B** and **End-Of-Token-1B**. Both the training objective (Cross-Entropy vs Distillation) as well as the tokenization scheme: **Tokens**, **Bytes**, **Bytes w/ <eot>** appear to influence the Average Error vs Validation BPB trends.

6.2 Results on Language Generation Benchmarks

We now visualize the Average top-1 error vs Validation BPB curves for the language generation tasks described in section 3.3. In addition, we also plot the averaged generation performance (in row 1).

1. We observe that for language generation as well, the same BPB can mean different averaged downstream performance for the three experiment categories - (**Token** , **Bytes** , **Bytes w/ <eot>**) as well as training objectives - supervised and distillation.

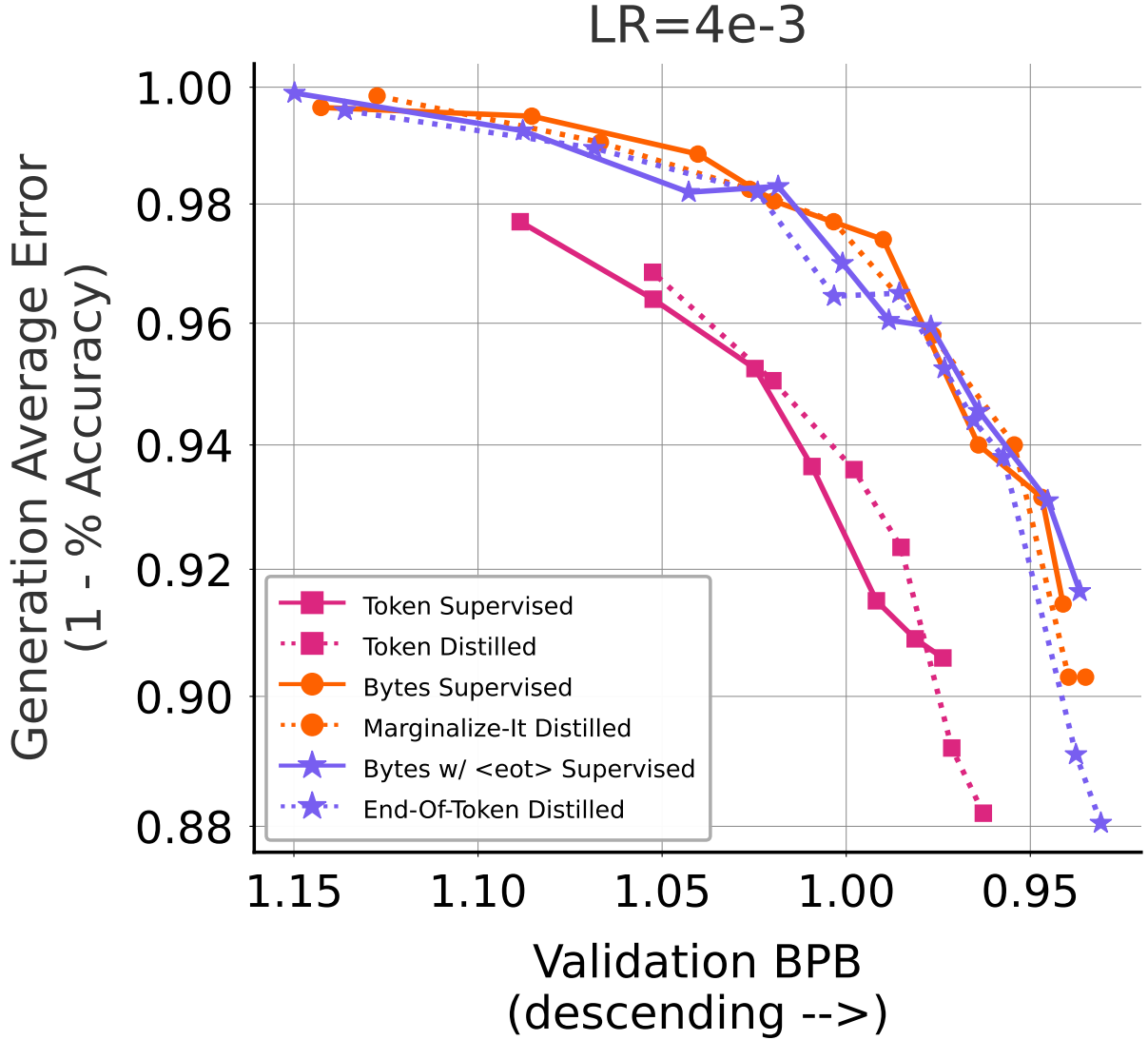


Figure 8: We plot the Average-top-1 error vs BPB curves for the MBPP and the NQ benchmarks. We notice an exponential decay for all three scenarios, with significantly different trends for the **Token-1B** vs **Bytes-1B** and **End-Of-Token-1B** models. Distilled models as well demonstrate slightly different scaling behavior compared to the supervised counterparts.

6.3 Results on Machine Translation Benchmarks

Lastly we plot BLEU score vs Validation BPB curves for Machine Translation tasks (Figure 9). We observe that:

1. The **Token-1B** models, **Bytes-1B** models and **End-Of-Token-1B** models show different scaling behaviors.
2. On average, **End-Of-Token** distillation appears to exhibit monotonic trend compared to the noisy **Bytes w/ <eot> Supervised** curves.

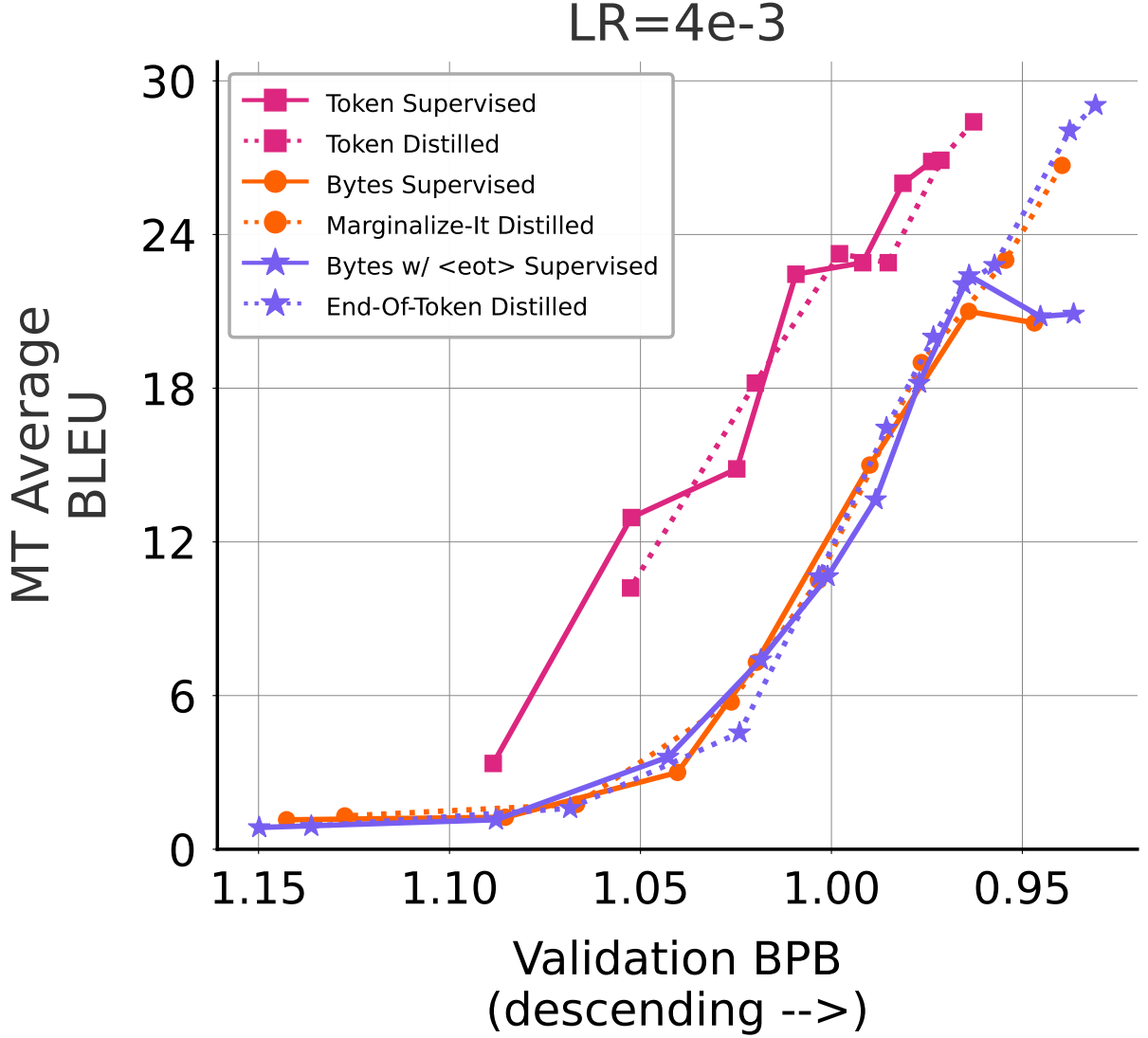


Figure 9: We plot averaged BLEU score vs Validation BPB performance, and on the averaged BLEU performance for the six scenarios for LR=4e-3. We notice the differences in the scaling behavior across tokenization schemes as well as the training objectives. **End-Of-Token** distillation appears to exhibit monotonic trend compared to the noisy supervised curves.

6.4 Fitting the Downstream Task Error vs Validation BPB Scaling Law

We fit the equation 2 for our six experiments on averaged top-1 error calculated over total 4 Multiple Choice and 2 language generation benchmarks (from 3.3) where y denotes the BPB value, $z(y)$ is the Averaged Top-1 Downstream Task Error, and ϵ , k and γ are fit using data. The scaling law equations are detailed in Table 4. The interpolations and extrapolations are visualized in Figure 10. The curves are bounded by the asymptotic BPB c , obtained from the power law equations in Table 2.

$$z(y) = \epsilon - k \cdot e^{-\gamma y} \quad (2)$$

Table 4: Fitted downstream error scaling laws $z(y) = \epsilon - k \cdot e^{-\gamma y}$ (equation 2) for each model variant and learning rate of 4e-3, where y is the validation BPB.

Method	Model Architecture	Scaling law equation $z(y)$
Token Supervised	Token-1B	$0.7501 - 934.3212 e^{-8.7782 y}$
Token Distilled	Token-1B	$0.7383 - 1292.0124 e^{-9.2132 y}$
Bytes Supervised	Bytes-1B	$0.7686 - 1231.1241 e^{-9.3532 y}$
Marginalize-It Distilled	Bytes-1B	$0.7721 - 1137.3004 e^{-9.2265 y}$
Bytes w/ <eot> Supervised	End-Of-Token-1B	$0.7972 - 185.6303 e^{-7.1822 y}$
End-Of-Token Distilled	End-Of-Token-1B	$0.7776 - 827.6076 e^{-8.8125 y}$

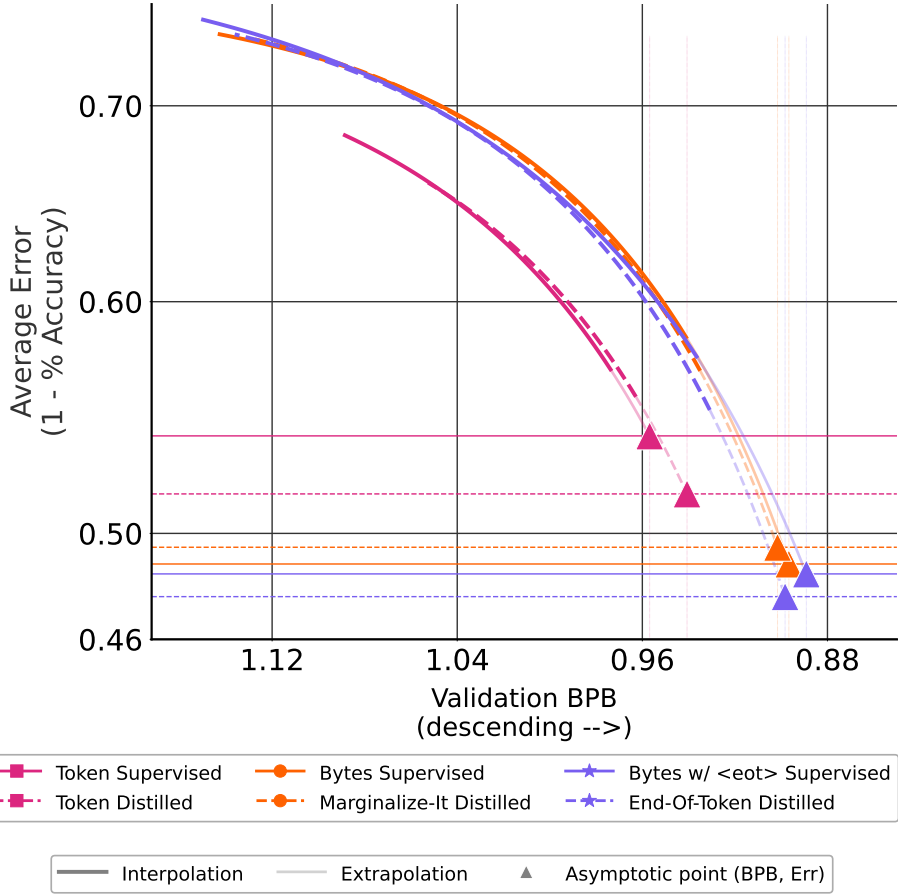


Figure 10: We fit the downstream error scaling laws for six benchmarks with respect to Validation BPB. All six experiments follow a smooth trend. **Token-1B** curves show significantly different behavior compared to the **Bytes-1B** and **End-Of-Token-1B** curves. The training objective (distillation vs supervised (Cross Entropy)) also affects the scaling laws. **End-Of-Token Distilled**, by preserving the teacher distribution exactly as well as with the highest amount of compute spent on fixed amount of data outperforms all of the other models asymptotically. **Byte** and **Bytes w/ <eot>** models, despite having fewer total parameters asymptotically surpass the **Token** models.

6.5 Downstream Task Performance Upper Bound

To calculate the asymptotic downstream task performance upper bound, we use the Average top-1 error at asymptotic validation BPB c , for each curve obtained from equation 2. We then compute the asymptotic

performance at that BPB as shown in Figure 10. Table 4 shows the asymptotic BPB as well as the task performance of the six experiments.

We observe that:

1. We notice that distilled **Token-1B** asymptotically outperforms over its supervised counterpart.
2. **End-Of-Token Distilled** method outperforms both **Token Distilled** and **Bytes w/ <eot> Supervised** asymptotically.
3. Distilled **Bytes-1B** performs on par or suboptimally compared to its supervised model potentially because it uses an approximation of the teacher distribution. It however still outperforms asymptotic distilled **Token-1B** accuracy.
4. Lastly, we note that by spending around $\approx 30.94\%$ additional compute on a fixed amount of data, **Bytes w/ <eot> Supervised** shows better asymptotic behavior compared to the **Bytes Supervised** model.
5. **End-Of-Token Distilled** shows better asymptotic downstream performance (lowest Average top-1 error) compared to the **Bytes w/ <eot> Supervised** model.

Table 5: Asymptotic validation BPB (c) and the corresponding downstream average accuracy $a^* = 1 - \%z^*$ where z^* is the Asymptotic Average top-1 error predicted by the scaling law equation: 2, $z^* = \epsilon - k \cdot e^{-\gamma c}$, where c is obtained from the power law equations in Table 4 for each model variant and learning rate.

Method	c	a^* (%)
Token Supervised	0.9568	46.0
Token Distilled	0.9407	48.4
Bytes Supervised	0.8967	51.2
Marginalize-It Distilled	0.9016	50.5
Bytes w/ <eot> Supervised	0.8891	51.6
End-Of-Token Distilled	0.8983	52.4

6.6 When Lower BPB Does Not (Always) Imply Better Models

Our observations show that BPB (Bits-Per-Byte) does not allow fair comparison across models with different training objectives (Distillation and Cross-Entropy) and tokenization schemes: **Tokens**, **Bytes**, **Bytes w/ <eot>**. We show that the relationship between the BPB across **Tokens**, **Bytes**, **Bytes w/ <eot>** is nuanced in the context of benchmark evaluations. These disparities potentially arise from the differences in probability distributions learned by spending more compute per unit data on a small amount of data vs small amount of compute spent per unit data on larger amount of data under a fixed compute budget. Although all of the **Bytes-1B** and **End-Of-Token-1B** models surpass the **Token Supervised** BPB training curve, they exhibit worse downstream task performance across benchmarks, falsifying observation 5 when the models being compared have different training objectives and tokenization schemes, emphasizing the importance of this final calibration step while determining the quality of a language model before real-world deployment. See appendix G.1 and section 10 for additional discussion.

7 The Case for **End-Of-Token** Logits

To understand if **End-Of-Token** logits are advantageous, it is essential to find out when they will if at all surpass the distilled token models. To achieve that we use the downstream error prediction scaling laws from 6.4 as shown below:

7.1 Feather Plot : Visualizing Iso-FLOP Point Connections

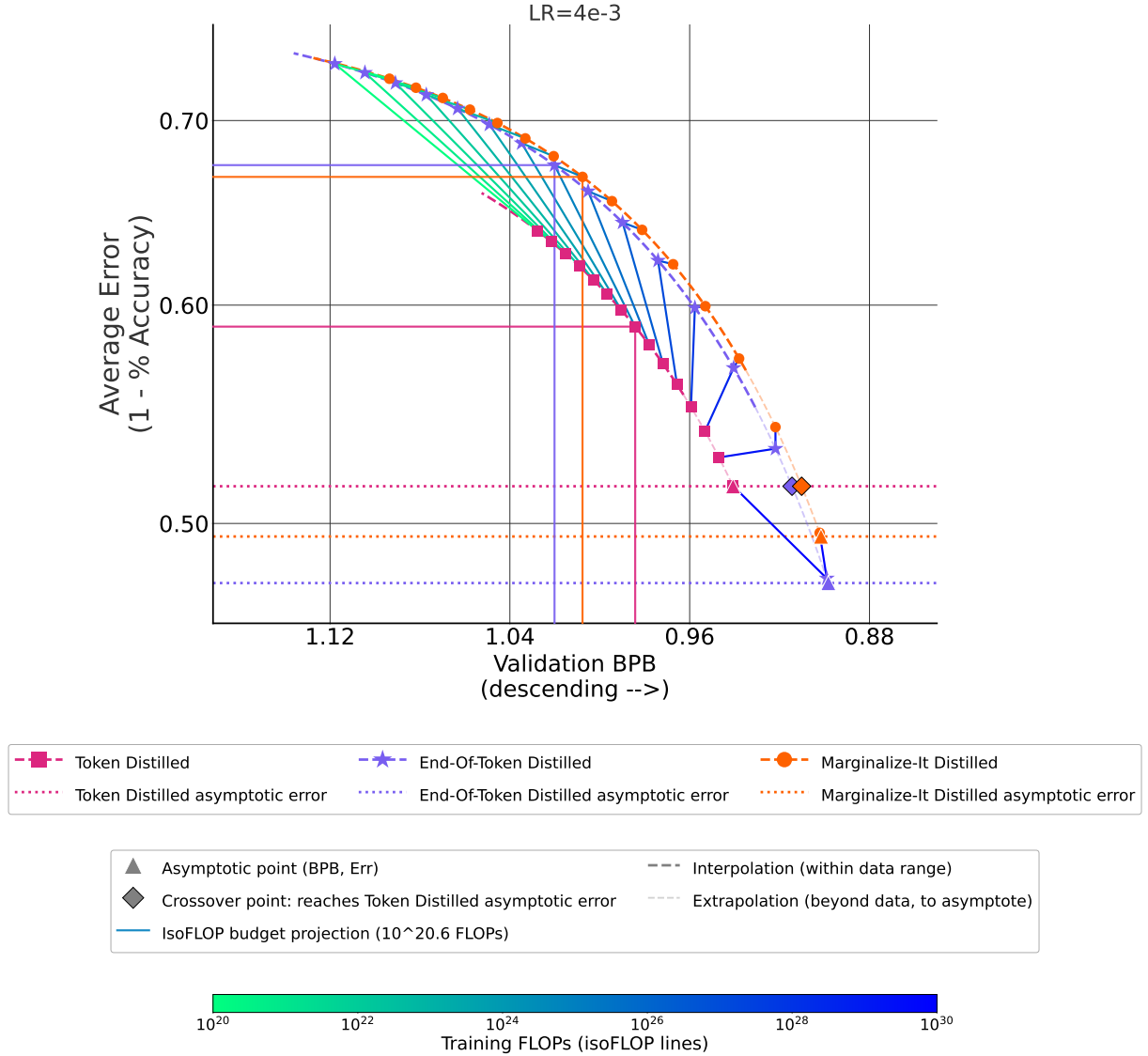


Figure 11: The figure shows a **Feather Plot** for a learning rate of $4e-3$. We show that **End-Of-Token Distilled** transformer model has the lowest Average top-1 downstream task performance error asymptotically compared to the **Marginalize-It** and **Token** distillation. We also visualize the Validation BPB values and the Average top-1 downstream task error at the compute budget of $10^{20.63}$ FLOPs (a randomly sampled value). We note the differences in observed BPB value and the task performance by projecting iso-FLOP points on the X and Y axes for the three curves. At that compute budget, **Token** distillation leads significantly, while both **End-Of-Token** distillation and **Marginalize-It** distillation lag behind.

We visualize lines connecting Iso-FLOP points which we call **Feather Plots** to understand how the BPB(Bits-Per-Byte) value as well as the downstream task performance evolves with compute budget. To plot them we sample points from FLOP range f_1 to f_2 . For each sampled point x we compute y using equation 1. Next we use the obtained validation loss in terms of BPB y to compute the downstream error $z(y)$ using equation 2. We connect the points obtained for three methods: **Token** , **Marginalize-It** , **End-Of-Token** distillation to analyze the trend. We observe for example the learnig rate of $4e-3$:

1. At low compute budgets, distilled **Token-1B** performance leads significantly compared to **Bytes-1B** and **End-Of-Token-1B** models.
2. As the compute budget progresses, distilled **End-Of-Token-1B** overtakes the **Bytes-1B** models.

- Eventually, distilled **End-Of-Token-1B** catches up with the **Bytes-1B** too and continues to improve until the asymptotic value of the performance.

7.2 Calculating Downstream Error Cross-Over Point with Token Distillation

Given the validation BPB prediction and downstream error prediction scaling laws for distilled **Token-1B**, **Bytes-1B**, **End-Of-Token-1B** models we want to find out at what compute budget **Marginalize-It Distilled** and **End-Of-Token Distilled** methods will give the same asymptotic downstream error prediction as that of a **Token Distilled** model.

Table 6: Training compute at which distilled **Bytes-1B** and **End-Of-Token-1B** match the asymptotic downstream performance of **Token-1B**. We take **Token Distilled** asymptotic BPB (the c term of its fitted BPB-FLOPs power law; column Token Dist. BPB) and map it through **Token Distilled** benchmark scaling law to obtain its asymptotic downstream accuracy (column Accuracy). For each byte method we then invert its own benchmark scaling law at that accuracy to get the BPB it must attain (column Method BPB), and invert its BPB-FLOPs power law at that BPB to get the required training compute (column Training FLOPs).

Method	Token Dist. BPB	Accuracy (%)	Method BPB	Method Training FLOPs
Marginalize-It Distilled	0.9407	48.4%	0.9102	2.04×10^{23}
End-Of-Token Distilled	0.9407	48.4%	0.9145	6.33×10^{22}

Table 7: Effective training-data volume at the compute budget required to achieve the **Token** distillation asymptotic performance from Table 5 for **Marginalize-It** and **End-Of-Token** methods. **Bytes** = Byte Units $\times$ Multiplier converts byte units into actual UTF-8 bytes of unique text, accounting for the fraction of each unit that is genuine content (Multiplier = 1 for **Marginalize-It Distilled**; = 4.5/5.5 for **End-Of-Token Distilled**, whose sequences carry an extra **<eot>** per 4.5 content bytes). **Data Savings** = Token to Byte/Bytes is how much less unique text the byte method needs to reach the same performance. To calculate storage savings we use our storage cost calculations from Table 10 where we equate the storage cost of one **Token** with 4.5 **Bytes** to obtain k (for top- k). **Storage Savings** = (Tokens $\times$ 4.5)/Byte Units compares the byte footprint of **Token Distilled** approach against the raw byte-unit count.

Metric	Formula	Marginalize-It Distilled	End-Of-Token Distilled
Training FLOPs	Compute to achieve Token distillation perf. (Table 5)	2.04×10^{23}	6.33×10^{22}
FPU _{bytes}	FLOPs per byte unit (Table 9)	8.81×10^9	9.44×10^9
FPU _{tokens}	FLOPs per token (Table 9)	8.18×10^9	8.18×10^9
Byte Units	Training FLOPs/FPU _{bytes}	2.32×10^{13}	6.70×10^{12}
Multiplier	(for actual data)	1	4.5/5.5
Bytes	Byte Units $\times$ Multiplier	2.32×10^{13}	5.48×10^{12}
Tokens	Training FLOPs/FPU _{tokens}	2.50×10^{13}	7.73×10^{12}
Token to Byte	Tokens $\times$ 4.5	1.12×10^{14}	3.48×10^{13}
Data Savings	Token to Byte/Bytes	4.85 $\times$	6.35 $\times$
Storage Savings	Tokens/(Byte Units/4.5)	4.85 $\times$	5.19 $\times$

7.3 Results

- We summarize our results in Table 7. Distilled **End-Of-Token-1B** training runs asymptotically outperform **Token-1B** model, hence we can compute the crossover points of **Bytes-1B** and **End-Of-Token-1B** models with **Token-1B** model.
- Since **End-Of-Token** uses the **<eot>** tokens every 4.5 bytes, the actual text data is less than the byte units used for training. Taking this into account, **End-Of-Token Byte Logits** can reduce the storage cost to approximately **one-fifth** compared to token distillation when trained for upto 6.7 Trillion byte units or data equivalent to 1.2 Trillion Llama 3-8B tokens.

End-Of-Token-1B demonstrates better asymptotic downstream performance compared to distilled **Token-1B**. It requires **one-sixth** text data and it can save storage costs upto **one-fifth**.

8 Comparisons with Open-Weight models

We now compare our asymptotic predictions for each of the six experiments described in section 3.2 with the open weight models: Llama 3.2-1B model (Meta AI, 2024), Gemma-3-1B-pt (Kamath et al., 2025), and Gemma 2B (Team et al., 2024) in Table 8. We evaluate these models using (Gao et al., 2024).

Table 8: Predicted asymptotic downstream accuracy (%) in the for our LR=4e-3 obtained from table 15, obtained by extrapolating the scaling trend. Comparisons with the open-weights model with our asymptotic predictions reveal our distilled **End-Of-Token-1B** model surpass the Meta AI (2024), Kamath et al. (2025), and Team et al. (2024) models on averaged downstream tasks by upto 6.5% and 8.1%, 2.1% respectively. Additional details for llama and gemma models are provided in table 18

Model	Method	Asymptotic Avg. Accuracy(%)
Token-1B	Token Supervised	46.0
Token-1B	Token Distilled	48.4
Bytes-1B	Bytes Supervised	51.2
Bytes-1B	Marginalize-It Distilled	50.5
End-Of-Token-1B	Bytes w/ <eot> Supervised	51.6
End-Of-Token-1B	End-Of-Token Distilled	52.4
Llama-3.2-1B	Meta AI (2024)	45.9 ($\approx$ 9T BPE tokens)
Gemma-3-1B-pt	Kamath et al. (2025)	44.3 ($\approx$ 2T BPE tokens)
Gemma 2b	Team et al. (2024)	50.3 ($\approx$ 3T BPE tokens)

9 Related Work

Scaling Laws with Overtraining. Which factors affect the power law exponent with overtraining has been a topic of great interest in deep learning. Kaplan et al. (2020) show that changing the architecture from transformers to LSTMs does not change the power law exponent for shorter contexts. (Bahri et al., 2024) show that the power law exponent changes with data distribution. (Henighan et al., 2020) show that the power law exponents change across modalities and also across image resolutions. (Bahri et al., 2024; Bansal et al., 2022) show that the noise level in data can also significantly affect the exponents. In the context of language modeling (Hestness et al., 2017) show that character models learn relationships between characters with fewer samples compared to word level language models learn from words. Several recent works focus on the overtraining of language models. (Sardana et al., 2023) demonstrate that when the inference budget is large enough, it is better to train a small model for longer. Gadre et al. (2024) derive scaling laws with overtraining for different model sizes and training datasets. Limisiewicz et al. (2026) analyze the impact of data compression on scaling laws. Lee et al. (2026) show that the autoregressive byte models approach performance parity with their BPE counterparts as compute scales. Distinct from prior work, we study power laws for $\approx$ fixed model size with overtraining along two axes: training objective and the tokenization scheme. We observe that both axes affect the exponent as seen from table (11) and table (15). We show that a simple addition of an <eot> token every $\approx$ 4.5 bytes can improve a model’s asymptotic downstream performance: an effect explained by the additional FLOPs spent for a fixed amount of data from the increased sequence length at fixed context.

Scaling Laws for Downstream Tasks. Schaeffer et al. (2023) argue that the capabilities that appear to be emergent in language models are due to the researcher’s choice of metric. In the context of transfer learning Isik et al. (2025) study how the choice of the pretraining data affects downstream cross entropy and Machine Translation performance (BLEU) score. Gadre et al. (2024) establish the relationship between the Average top-1 downstream error across a number of benchmarks and the validation Cross-Entropy Loss. They demonstrate that using validation loss alone while comparing models

trained on different data distributions can be misleading. Owen (2024) show that the aggregate benchmark performance is decently predictable, though predicting individual performance is harder. Bhagia et al. (2024) establish a two stage approach where they use N (# of parameters) and D (# of tokens) to predict the intermediate task loss which is then mapped to the task performance. Lourie et al. (2025) conduct a meta-analysis of the downstream tasks considered in Gadre et al. (2024) to demonstrate that the task-specific scaling trends may vary. They further compare them with the findings from Magnusson et al. (2025) to show that the scaling behavior depends on the experimental setup and that the choice pretraining corpus, validation corpus, or downstream task can affect the scaling laws. Our work studies how the training objective and the tokenization scheme can affect the downstream performance with Training FLOPs.

Pretraining Distillation. Hinton et al. (2015) introduced distillation and showed that a small model can be improved by using the output probabilities of a large model. Busbridge et al. (2025) build scaling laws to determine the optimal compute allocation between student and the teacher. Peng et al. (2025) investigate the effect of different design decisions in token to token distillation. Beyond language modeling Beyer et al. (2022) proposed the “patient and consistent teacher” hypothesis for ResNet-50 models (He et al., 2016) trained on Imagenet (Deng et al., 2009). In this work we study pretraining distillation for both token and byte models.

Token to Bytes transfer. Minixhofer et al. (2025) present an approach for adapting pretrained token-level models to operate on raw bytes. Minixhofer et al. (2026) and Bao et al. (2026) propose methods for distilling token models into byte models in the fine-tuning stage. Hayase et al. (2025) and Phan et al. (2024) study the exact BPE to bytes transfer to primarily tackle the **prompt boundary problem** where the prompt, when ends in the middle of a BPE token, results into distorted next token distributions at inference time. These methods can be used to convert token logits to byte logits. However they remain computationally expensive and require multiple inference passes over the teacher to obtain exact conversions. We circumvent this problem by simply marginalizing the prefix-matched token distributions in **Marginalize-It** method and introducing an additional `<eot>` token in the vocabulary and the sequences during training to preserve the distribution in the **End-Of-Token** method.

Byte Level Models. Al-Rfou et al. (2018), Choe et al. (2019), El Boukkouri et al. (2020) laid the groundwork for byte level language modeling. J. H. Clark et al. (2022) introduce CANINE, an architecture with downsampling, followed by a deep transformer stack and upsampling layers for efficient character level modeling. Xue et al. (2022) train an encoder-decoder model operating on UTF-8 bytes and show that byte level transformers are data efficient learners, though take longer to train and are much expensive during inference time. Yu et al. (2023) introduce the concept of patching to enable sub-quadratic self attention. Nawrot et al. (2023) propose a hierarchical architecture for byte level modeling jointly performs language modeling and token segmentation. Pagnoni et al. (2025) propose a new architecture with a local encoder and decoder that operates on patches of raw bytes. Hwang et al. (2026) propose dynamic chunking for end to end byte level sequence modeling. Very recently EvaByte Zheng et al. (2025) trained byte level transformers with various tricks like multi-byte prediction and linearized attention. We study three decoder-only transformer variants (Vaswani et al., 2017) with vocabulary sizes of 128256 (**Token-1B**), 260 (**Bytes-1B**), and 261 (**End-Of-Token-1B**). They differ in both parameter count (from the embedding layers) and FLOPs (by varying sequence lengths). By preserving the teacher distribution exactly while spending $\approx 30.94\%$ additional compute on a fixed amount of data, the **End-Of-Token-1B** model asymptotically outperforms the distilled **Token-1B**. We also note that **End-Of-Token-1B** model despite having fewer parameters can asymptotically surpass **Token-1B** on downstream tasks.

10 Discussion

Marginalize-It vs End Of Token. Our findings show that **Marginalize-It**, despite being an approximation asymptotically outperforms the **Token** distillation baseline. **End-Of-Token** offers two advantages: 1) it preserves the teacher distribution exactly, and 2) it improves asymptotic performance of the supervised **End-Of-Token-1B** baseline itself at the cost of an extra $\approx 30.94\%$ compute per unit of data. Due to these two properties, **End-Of-Token** distillation asymptotically outperforms all of the other five models on downstream tasks.

The role of <eot> tokens on power laws. Our work shows that simply adding <eot> token every ≈ 4.5 bytes on average affects the power law coefficients 3. We also observe that supervised **End-Of-Token-1B** yields better asymptotic downstream performance compared to the **Bytes-1B** baseline. This finding opens many exciting avenues for future work.

Inference Costs. While our scaling laws show that, distilling Llama 3-8B (Grattafiori et al., 2024) model into **End-Of-Token-1B** models with 1.28B total parameters is asymptotically better than distilling into **Token-1B** models with 1.81B total parameters, **End-Of-Token-1B** models remain significantly expensive at inference time. Comparing the asymptotic performance of a larger **Token-1B** model against that of a smaller **End-Of-Token-1B** transformers with matched inference cost could reveal whether distilled **End-Of-Token-1B** transformers are also efficient at inference time; this is beyond the scope of this study.

BPB vs. Benchmark Performance Discrepancy. Our observations reveal that BPB is not a metric that enables fair comparison across models with different training objectives and tokenization schemes. Many other works observe the same phenomenon for different optimizers (Zhang et al., 2026), data distributions (Gadre et al., 2024), long-context setting, (Fang et al., 2025), byte modeling (Lee et al., 2026), and in theoretical analyses (Liu et al., 2023; Velićković et al., 2026). We identify additional settings in which both the training objective (Cross Entropy Supervised vs Distillation) and the tokenization scheme (**Token** , **Bytes** , **Bytes w/ <eot>**) can alter the downstream-task-performance vs. validation-BPB scaling trajectories, implying that a model with a lower BPB may not be necessarily a better model.

10.1 Future work

Our scaling study focuses on dense transformer models. It would be interesting to explore how introducing a third axis of sparsity, by training Mixture-of-Experts models affects these power laws. Next, our study examines scaling trends across varying data budgets with approximately fixed model size of 1.28 billion layer parameters, ranging up to 1 trillion bytes. We also note that some of our larger training runs epoch. Future work should explore training with larger model sizes and more data. Next we report our findings based on eight benchmarks spanning three categories: Multiple Choice Question Answering, Language Generation, and Machine Translation. Future work should explore evaluating models on more benchmarks. Lastly, understanding the training dynamics of distilling large byte transformers into smaller ones remains an open question.

Acknowledgments

We thank Jonathan Hayase, Alisa Liu, and Benjamin Minixhofer for thoughtful discussions and feedback on this work.

Contents

1	Introduction	1
1.1	Summary of Contributions	3
2	Logit Conversion Methods	4
2.1	Marginalize-It Logits	4
2.2	End-Of-Token Logits	4
3	Scaling Study Design	6
3.1	Experimental Setup	6
3.2	Scaling Along Two Axes: Tokenization and Training Objectives	6
3.3	Benchmark Evaluations	6
4	Scaling Trends I: Validation BPB vs Training FLOPS	7
4.1	Validation BPB vs Training FLOP Plots	7
5	Scaling Trends II: Task Performance vs Training FLOPs	8
5.1	Results on Multiple Choice Question Answering Benchmarks	9
5.2	Results on Language Generation Benchmarks	9
5.3	Results on Machine Translation Benchmarks	10
5.4	Concluding Remarks	11
6	Scaling Trends III: Task Performance vs Validation BPB	11
6.1	Results on Multiple Choice Question Answering Benchmarks	12
6.2	Results on Language Generation Benchmarks	12
6.3	Results on Machine Translation Benchmarks	13
6.4	Fitting the Downstream Task Error vs Validation BPB Scaling Law	14
6.5	Downstream Task Performance Upper Bound	15
6.6	When Lower BPB Does Not (Always) Imply Better Models	16
7	The Case for End-Of-Token Logits	16
7.1	Feather Plot : Visualizing Iso-FLOP Point Connections	17
7.2	Calculating Downstream Error Cross-Over Point with Token Distillation	18
7.3	Results	18
8	Comparisons with Open-Weight models	19
9	Related Work	19
10	Discussion	20
10.1	Future work	21
A	Experimental Setup Details	24
A.1	Model Architecture and FLOP Calculations.	24
A.2	Pretraining Dataset, Tokenization, and Data Scales	24
A.3	Loss Function and Optimization Settings	24
A.4	Sequence Length and Global Batch Size.	25
A.5	Bits Per Byte Calculation.	25
A.6	Storage Cost Calculations of Token and Byte Logits	25
B	Additional Results using Validation Dataset 1	27
C	Comparisons with open-weight models: Additional details	37
D	On Scaling Laws Sensitivity	38
D.1	Leave One Out Error Analysis	38
D.2	Residual Weighting Scheme Sensitivity Analysis	39
D.3	Choice of the validation data	40

E	Additional Results using Validation Dataset 2	42
E.1	BPB vs Training FLOPs power law equations	42
E.2	Average top-1 Error Scaling Laws	42
E.3	Feather Plots : Comparisons using validation dataset 2	44
E.4	Predicting the asymptotic Average Downstream task performance	44
F	Distillation Methods: Additional Details	46
F.1	Marginalize-It and End-Of-Token Pseudocode	46
F.2	Comparing Marginalize-It method with End-Of-Token	46
G	On BPB vs Benchmark Performance Discrepancy	49
G.1	Why the same BPB may imply different downstream performance?	49

A Experimental Setup Details

In this section we provide additional details on the design decisions for our experiments.

A.1 Model Architecture and FLOP Calculations.

To train both the token and byte student models, we use the transformer (Vaswani et al., 2017) architecture with 1 billion parameters. We use SwiGLU (Shazeer, 2020) activation function. The differences in the number of parameters and training FLOPs arise from the differences in the vocabulary size – byte models use fewer parameters and FLOPs per byte.

We use Llama3-8B (Grattafiori et al., 2024) as our teacher model.

Property	Formula	Token-1B	Bytes-1B	End-Of-Token-1B
Model Dimension	d	2048	2048	2048
FFN dimension	$d_{ffn} = \lceil 4 \cdot \frac{2}{3} \cdot \frac{d}{256} \rceil \cdot 256$	5632	5632	5632
Number of Layers	n	25	25	25
Number of Heads	n_{heads}	16	16	16
Vocabulary size	V	128256	260	261
Attention Parameters per Layer	$N_{attn} = 4 \cdot d^2$	16777216	16777216	16777216
FFN Parameters per Layer	$N_{ffn} = 3 \cdot d \cdot d_{ffn}$	34603008	34603008	34603008
RMSNorm Parameters per Layer	$N_{norm} = 2 \cdot d$	4096	4096	4096
Total Layer Parameters	$N_{layers} = n \cdot (N_{attn} + N_{ffn} + N_{norm})$	1284608000	1284608000	1284608000
Pre Output Norm Parameters	$N_{preoutputnorm} = d$	2048	2048	2048
Total Non-Embedding Parameters	$N_{nonembed} = N_{layers} + N_{preoutputnorm} + d \cdot V$	1547278336	1285142528	1285144576
Total Parameters	$N_{total} = N_{nonembed} + d \cdot V$	1809946624	1285675008	1285679104
Sequence Length	l	2048	9216	11264
Self Attention FLOPs	$F_{attn} = 4 \cdot n \cdot (d/n_{heads}) \cdot n_{heads} \cdot \frac{l+1}{2}$	209817600	943820800	1153536000
Self Attention QKVO FLOPs	$F_{qkvo} = (2+2) \cdot 2 \cdot n \cdot d^2$	838860800	838860800	838860800
Feed Forward FLOPs	$F_{ffn} = 2 \cdot n \cdot 2 \cdot d \cdot d_{ffn}$	1153433600	1153433600	1153433600
De Embedding FLOPs	$F_{deemb} = 2 \cdot d \cdot V $	525336576	1064960	1069056
Total FLOPs per unit	$F_{total} = 3 \cdot (F_{attn} + F_{qkvo} + F_{ffn} + F_{deemb})$	8182345728	8811540480	9440698368

Table 9: Model Architecture Comparison

A.2 Pretraining Dataset, Tokenization, and Data Scales

We use the Llama-2 (Touvron et al., 2023) pretraining mixture for all of our experiments. We tokenize the data using two tokenizers: BPE-Tiktoken (Llama3-8B) tokenizer and the byte tokenizer. Our byte tokenizer uses a vocabulary size of 260: 256 UTF-8 bytes and 4 special tokens. We train multiple 1B models with varying D where D is the number of tokens/byte units.

A.3 Loss Function and Optimization Settings

A typical knowledge distillation setup involves two components: αL_{KL} , a loss term computed using the teacher’s logits and $(1 - \alpha)L_{CE}$, the next token prediction loss computed against the ground truth data. When $\alpha = 0$, the objective reduces to supervised training (Cross-Entropy loss with respect to data) and when $\alpha = 1$, the model is trained purely via teacher distillation. We use equation 3 as our loss function and use α , the mixing coefficient to control our supervised and distillation runs.

$$L_{KD} = \alpha \cdot L_{KL} + (1 - \alpha) \cdot L_{CE} \quad (3)$$

$$L_{KL} = \frac{1}{n} \sum_{t=1}^n \sum_c p_T(c | x_{<t}) \log \frac{p_T(c | x_{<t})}{p_S(c | x_{<t})} \quad (4)$$

$$L_{CE} = -\frac{1}{n} \sum_{t=1}^n \sum_c y_{t,c} \log p_S(c | x_{<t}) \quad (5)$$

where:

- t is the position index in the sequence

- n is the total number of tokens in the sequence
- c is the token index over the vocabulary V
- $y_{t,c} \in \{0, 1\}$ is a one-hot indicator that the ground-truth token at position t is c
- $p_T(c \mid x_{<t})$ is the teacher model’s predicted probability distribution over the next token given all preceding tokens $x_{<t}$
- $p_S(c \mid x_{<t})$ is the student model’s predicted probability distribution over the next token given all preceding tokens $x_{<t}$
- $\alpha \in [0, 1]$ is a weighting factor balancing the KL divergence and cross-entropy losses

We train our models using the AdamW (Loshchilov & Hutter, 2017) optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ and a cosine learning rate schedule. We set warmup to 10% of the maximum number of steps for each data scale use and three peak learning rates $\in \{1\text{e-}3, 4\text{e-}3, 8\text{e-}3\}$. We use independent weight decay (Kosson et al., 2025) of $1\text{e-}4$ and a grad clip norm of 1.0.

A.4 Sequence Length and Global Batch Size.

We set sequence length to 2048 for our **Token-1B** models. We train models using 64 H200s per training run and use a global batch size is 0.5M tokens. For training byte transformers, our initial explorations with the exact same context and data ($8\text{K} \times 4 \times 64 = 2\text{M}$ bytes) per optimization step resulted in loss spikes. To stabilize the runs we use qk-norm and set the sequence length to 9216 for **Bytes-1B** and 11264 for **End-Of-Token-1B**, and maintain the global batch size as that of the token models. If the byte sequence formed by the 2048 tokens exceeds the byte sequence length, we truncate it; if the sequence is shorter, we pad it with padding tokens to preserve the context length. We use 64 H200s for each **Bytes-1B** and **End-Of-Token-1B** as well, maintaining the global batch size of $\approx 0.5\text{M}$ byte units as that of the token models. Hence, overall our byte models use much less data compared to token based models for the same compute budget.

A.5 Bits Per Byte Calculation.

We use BPB (Bits per Byte) (equation 6), a smooth tokenizer independent metric on a held out validation set to plot the scaling curves. The BPB is defined as the total cross-entropy loss $\mathcal{L}_{\text{CE}}$ summed over the validation data x divided by the total number of bytes B in x and scaled by a constant. Intuitively, it captures the average number of bits a model uses to represent the data.

$$\text{BPB} = \frac{L_{\text{CE}}(x) \cdot \log_2 e}{B} \quad (6)$$

Finally, We use Llama 3-8B (Grattafiori et al., 2024) and Llama3.2-1B (Meta AI, 2024) models as reference points to compare with our token and byte asymptotes. Since these models are overtrained on several trillion tokens, they simulate the infinite data regimes.

A.6 Storage Cost Calculations of Token and Byte Logits

The biggest pain point of using offline distillation to train small language models is the storage cost of the teacher logits. Each BPE token of a Llama3-8B model, for instance, produces 128256 logit values. A billion parameter model overtrained to 2T tokens, for instance, would require and $2T \times 4 \times 128256$ bytes $= 1.026048$ Exabytes assuming `float32` datatype. To circumvent this storage problem, researchers store only the *top-k* logits per token, where k typically is of the order of several hundreds. On the contrary, byte transformers use a fixed vocabulary of only ≈ 260 bytes, trading vocabulary size for increased sequence length. Our controlled study is designed to characterize this tradeoff.

In table 10 we detail the storage cost of storing one BPE token and their *top-k* logits. Note that for BPE tokens, we use the sparse representation of logits while for bytes logits we store the full distribution. By equating the total cost of BPE-tiktoken with bytes, we solve for $k = 4685/8$, which turns out to be

Table 10: Storage cost analysis for top-k logits per token. We solve for k and found it to be ≈ 600

Token type	Data Attribute	Datatype	Bytes per value	# Values	Total Cost
BPE-tiktoken	Token Logit values	float32	4	k	$4k + 4k + 4$
	Token Logit indices	int32	4	k	
	Tokens	int32	4	1	
Bytes	Byte Logits	float32	4	260×4.5	$4680 + 9$
	Bytes	int16	2	4.5	

$585.625 \approx 600$. Therefore for our controlled comparison we store 600 logits per BPE-tiktoken token and all of the byte logits i.e. 260 values for distillation.

B Additional Results using Validation Dataset 1

In this section we provide additional results for the learning rates of 1e-3 and 8e-3 for validation dataset 1 (which is used for analysis in the paper).

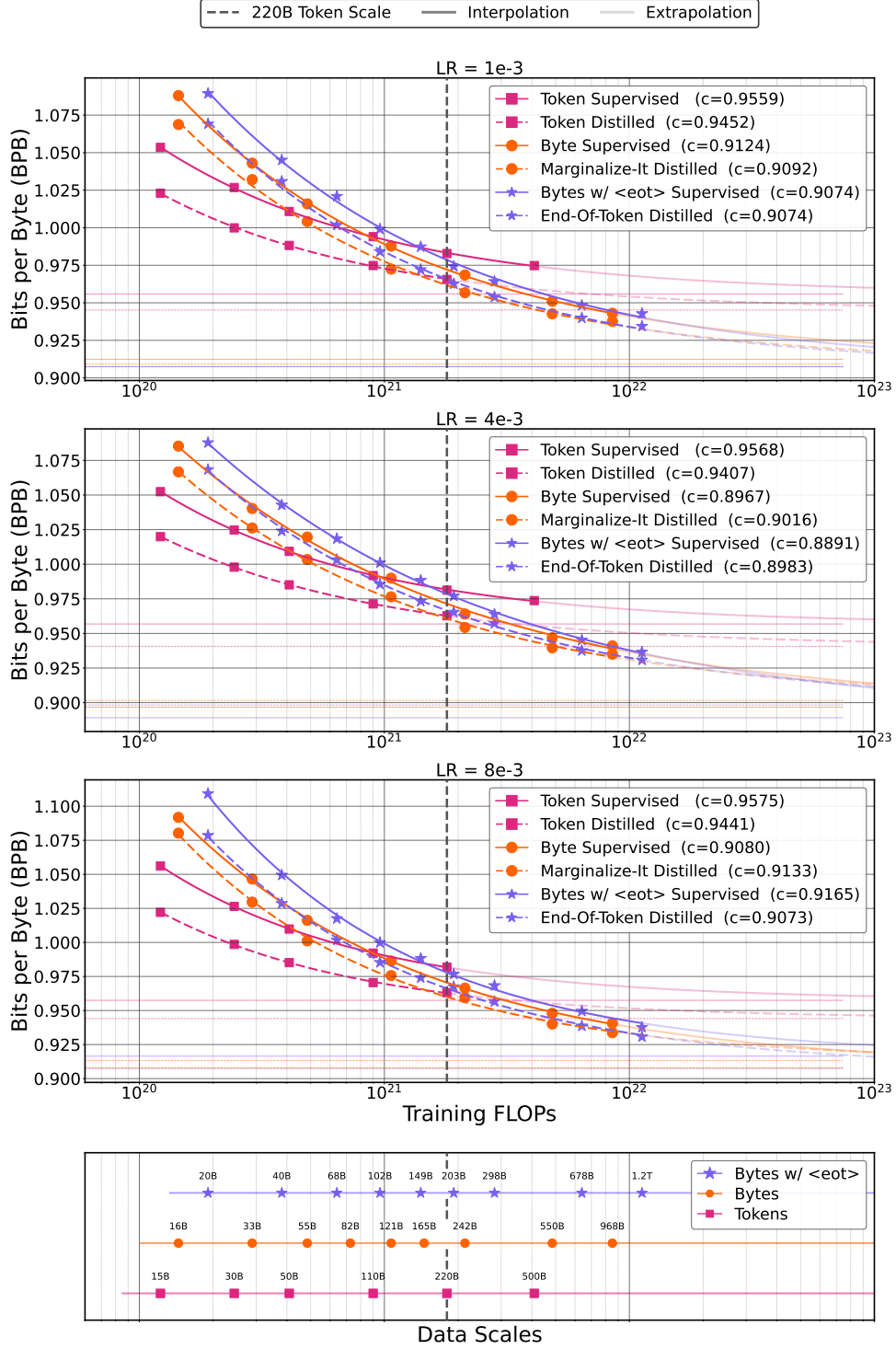


Figure 12: The figure shows our scaling curves for different data scales and learning rates for a fixed parameter count of approximately 1 billion parameter transformer models (see Table 9 for details). The distilled models are trained using Llama 3-8B as a teacher model. The validation BPB of the teacher model is 0.85511.

Table 11: Power-law equations $y = b \cdot x^a + c$ fitted to BPB vs. training FLOPs.

LR	Scenario	Equation
1×10^{-3}	Token Supervised	$y = 3.013 \times 10^8 \cdot x^{-0.4724} + 0.9559$
	Token Distilled	$y = 5.677 \times 10^8 \cdot x^{-0.4911} + 0.9452$
	Bytes Supervised	$y = 7.253 \times 10^7 \cdot x^{-0.4274} + 0.9124$
	Marginalize-It Distilled	$y = 1.503 \times 10^8 \cdot x^{-0.4449} + 0.9092$
	Bytes w/ <eot> Supervised	$y = 6.329 \times 10^7 \cdot x^{-0.4210} + 0.9074$
	End-Of-Token Distilled	$y = 3.383 \times 10^8 \cdot x^{-0.4593} + 0.9074$
4×10^{-3}	Token Supervised	$y = 1.075 \times 10^9 \cdot x^{-0.5004} + 0.9568$
	Token Distilled	$y = 2.803 \times 10^8 \cdot x^{-0.4754} + 0.9407$
	Bytes Supervised	$y = 4.467 \times 10^6 \cdot x^{-0.3658} + 0.8967$
	Marginalize-It Distilled	$y = 2.706 \times 10^7 \cdot x^{-0.4074} + 0.9016$
	Bytes w/ <eot> Supervised	$y = 3.080 \times 10^6 \cdot x^{-0.3546} + 0.8891$
	End-Of-Token Distilled	$y = 2.734 \times 10^7 \cdot x^{-0.4047} + 0.8983$
8×10^{-3}	Token Supervised	$y = 2.927 \times 10^9 \cdot x^{-0.5214} + 0.9575$
	Token Distilled	$y = 3.692 \times 10^9 \cdot x^{-0.5314} + 0.9441$
	Bytes Supervised	$y = 8.176 \times 10^7 \cdot x^{-0.4289} + 0.9080$
	Marginalize-It Distilled	$y = 2.271 \times 10^9 \cdot x^{-0.5028} + 0.9133$
	Bytes w/ <eot> Supervised	$y = 3.627 \times 10^9 \cdot x^{-0.5068} + 0.9165$
	End-Of-Token Distilled	$y = 6.619 \times 10^8 \cdot x^{-0.4729} + 0.9073$

Table 12: We demonstrate that the amount of FLOPs spent on a fixed amount of data (one BPE token in this case) affects the asymptotic BPB. Overall: Bytes w/ <eot> Supervised BPB < Bytes Supervised BPB < Token Supervised BPB. and End-Of-Token Distilled BPB < Marginalize-It Distilled) BPB < Token Distilled BPB. We also note that for all learning rates, within the groups (based on training objective), the supervised and distilled models exhibit small differences in the asymptotic value. The FLOPs/Unit values are obtained from Table 9.

Method	FLOPs / Unit	Units	Total FLOPs Per Token	Asymptotic BPB		
				1e-3	4e-3	8e-3
Token Supervised	8.18×10^9	1.0	8.18×10^9	0.9559	0.9568	0.9575
Bytes Supervised	8.81×10^9	4.5	39.65×10^9	0.9124	0.8967	0.9080
Bytes w/ <eot> Supervised	9.44×10^9	5.5	51.92×10^9	0.9074	0.8891	0.9165
Token Distilled	8.18×10^9	1.0	8.18×10^9	0.9452	0.9407	0.9441
Marginalize-It Distilled	8.81×10^9	4.5	39.65×10^9	0.9092	0.9016	0.9133
End-Of-Token Distilled	9.44×10^9	5.5	51.92×10^9	0.9074	0.8983	0.9073

Table 13: Scaling-law goodness of fit. Each entry is the coefficient of determination $R^2 = 1 - \text{SS}_{\text{res}}/\text{SS}_{\text{tot}}$ for the fit $y = b \cdot x^a + c$ of BPB versus training FLOPs.

Scenario	R^2 (LR=1e-3)	R^2 (LR=4e-3)	R^2 (LR=8e-3)
Token Supervised	0.9999	0.9999	1.0000
Token Distilled	0.9996	1.0000	0.9997
Bytes Supervised	0.9999	0.9980	0.9996
Marginalize-It Distilled	0.9974	0.9989	0.9985
Bytes w/ <eot> Supervised	0.9982	0.9995	0.9984
End-Of-Token Distilled	0.9980	0.9990	0.9987

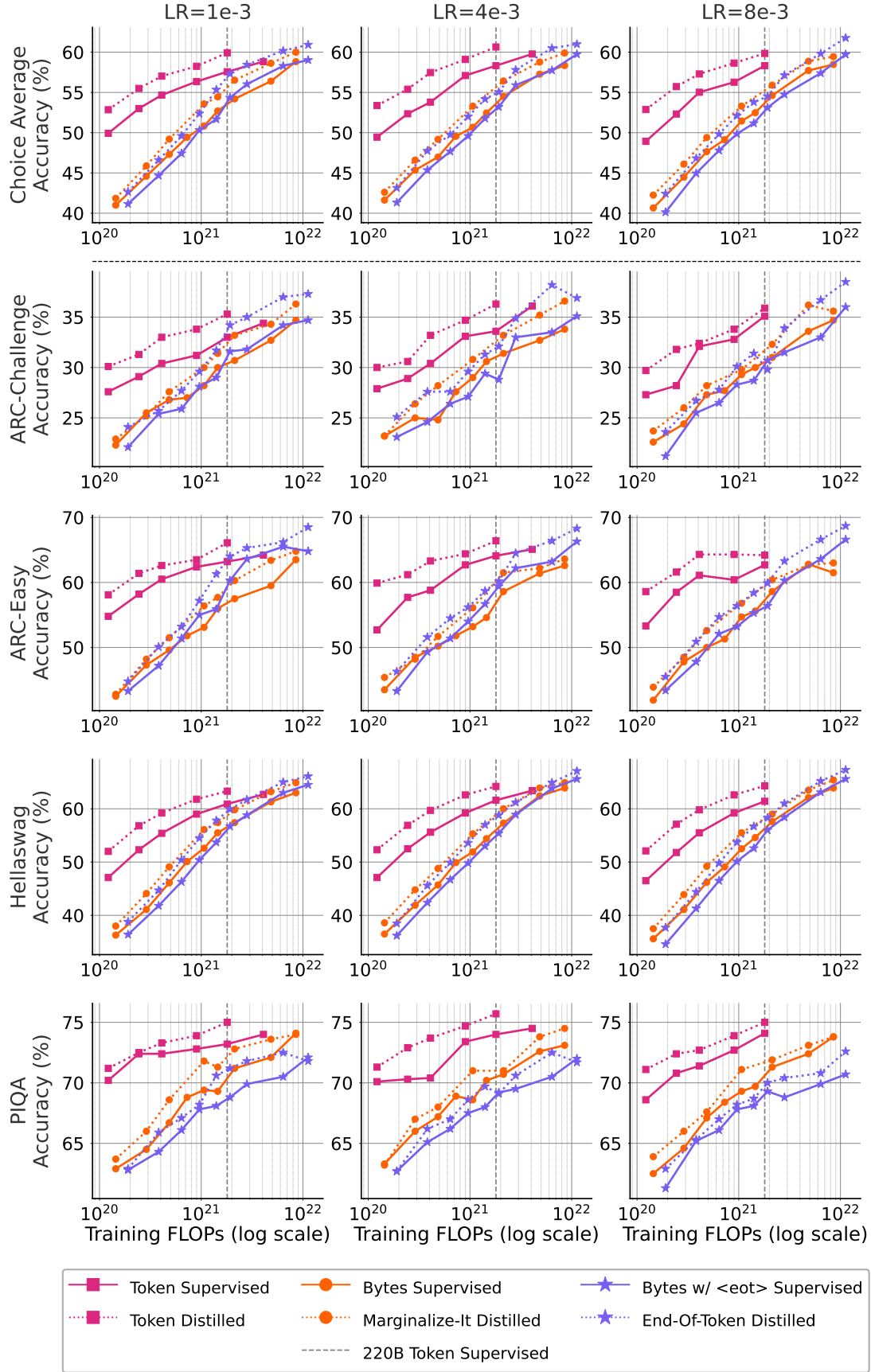


Figure 13: We plot Accuracy vs training FLOP curves for all the six scenarios from section 3.2. For all of the Multiple Choice Question Answering Benchmarks, we observe that the token models show better performance at low compute budgets and eventually plateau. Byte models, start worse yet improve at a much faster rate.

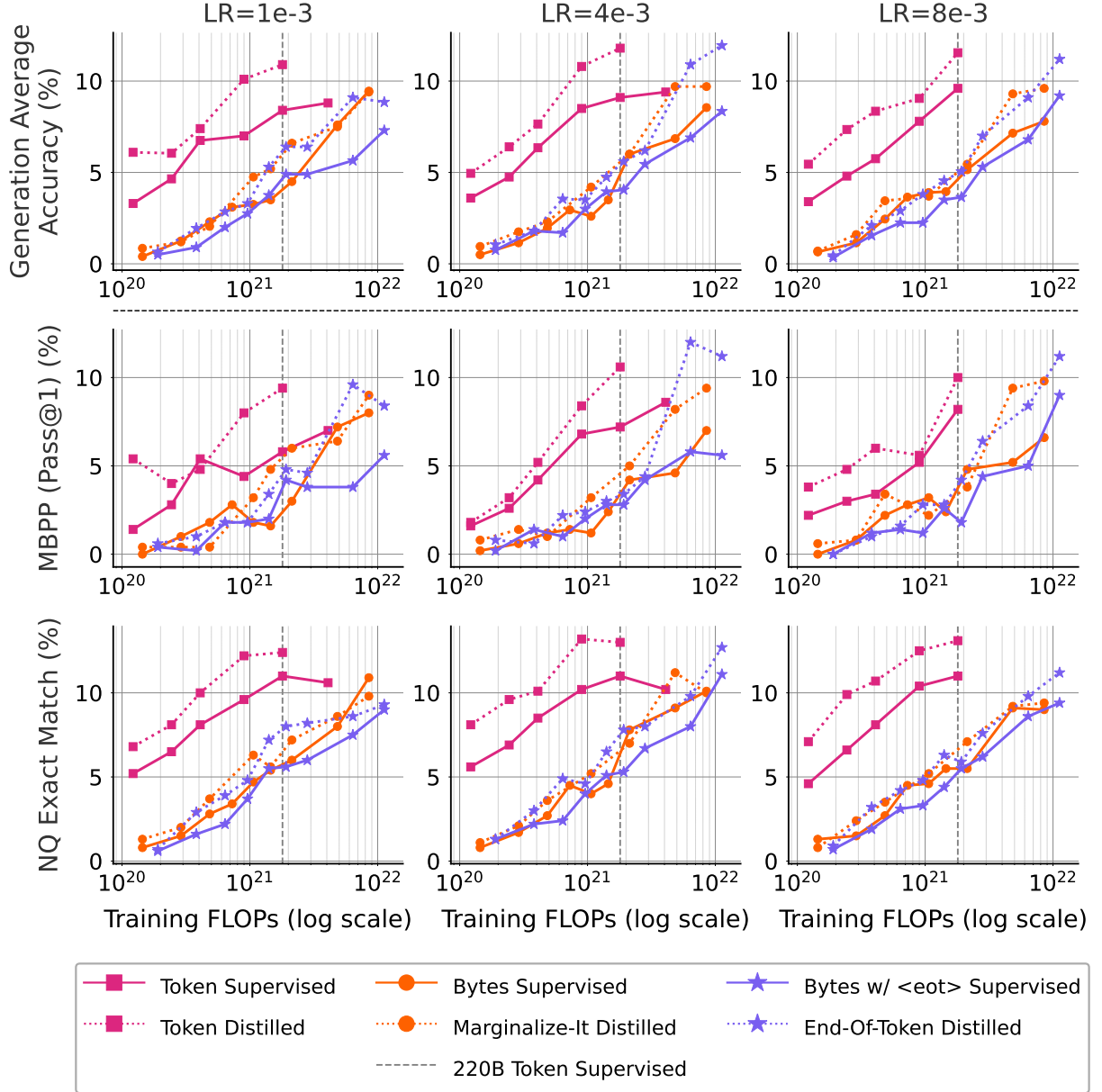


Figure 14: We visualize % Accuracy vs training FLOPs curves for language generation tasks. On generative tasks, token models achieve better performance at lower compute budgets and appear to plateau with the training FLOPs. **Bytes-1B** and **End-Of-Token-1B** models show a different scaling behavior, with much worse performance at a lower compute budget which continues to improve with FLOPs with no sign of saturation.

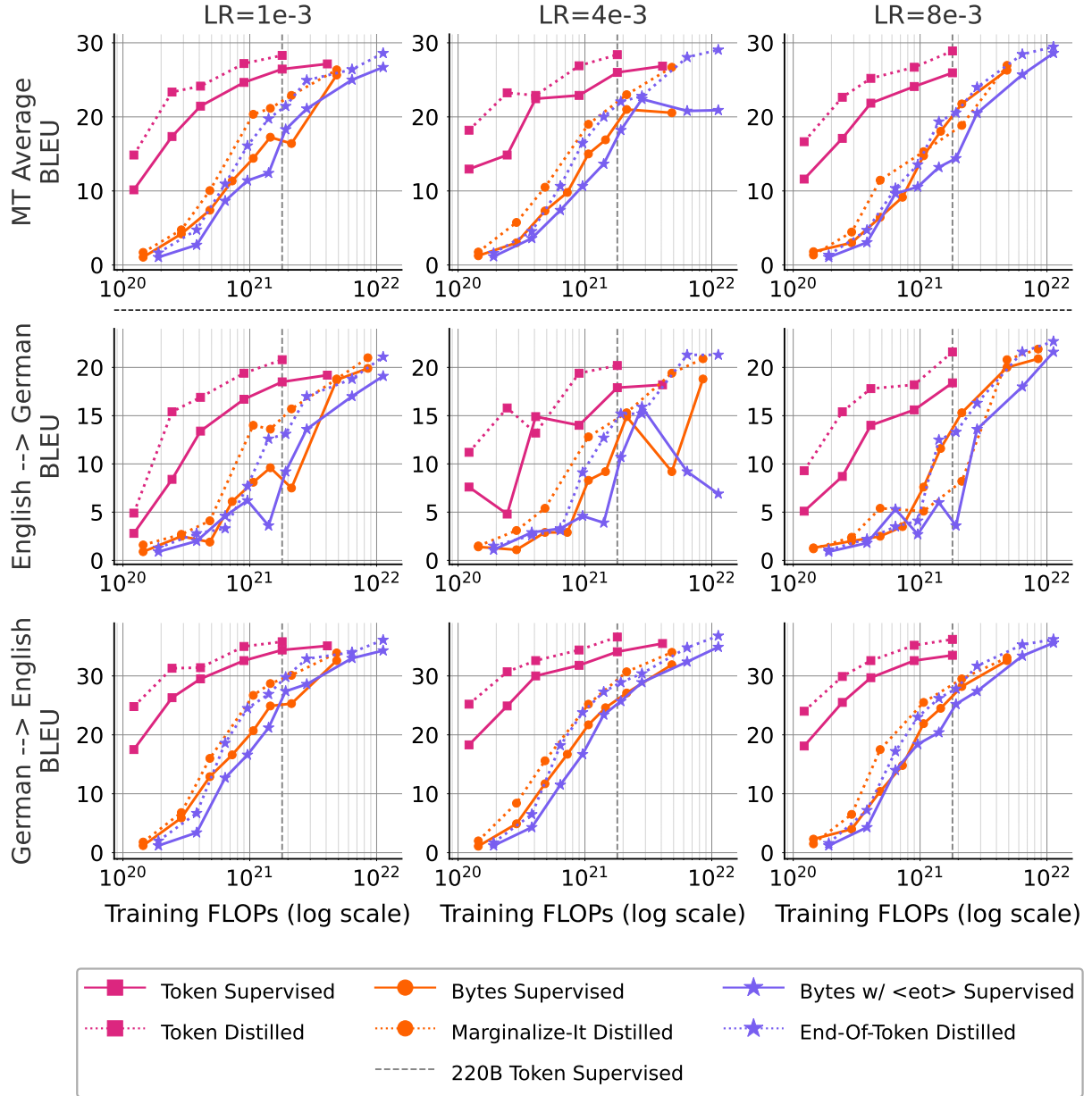


Figure 15: We plot BLEU score for the translation tasks for different training budgets. We observe clearly different scaling behaviors for token and byte models. Token models can achieve significantly better performance at a smaller compute budgets. The BLEU score appears to saturate with overtraining. Byte models both with and without the `<eot>` demonstrate extremely low BLEU scores at lower compute budgets yet the rate of improvement with overtraining is better compared to token models.

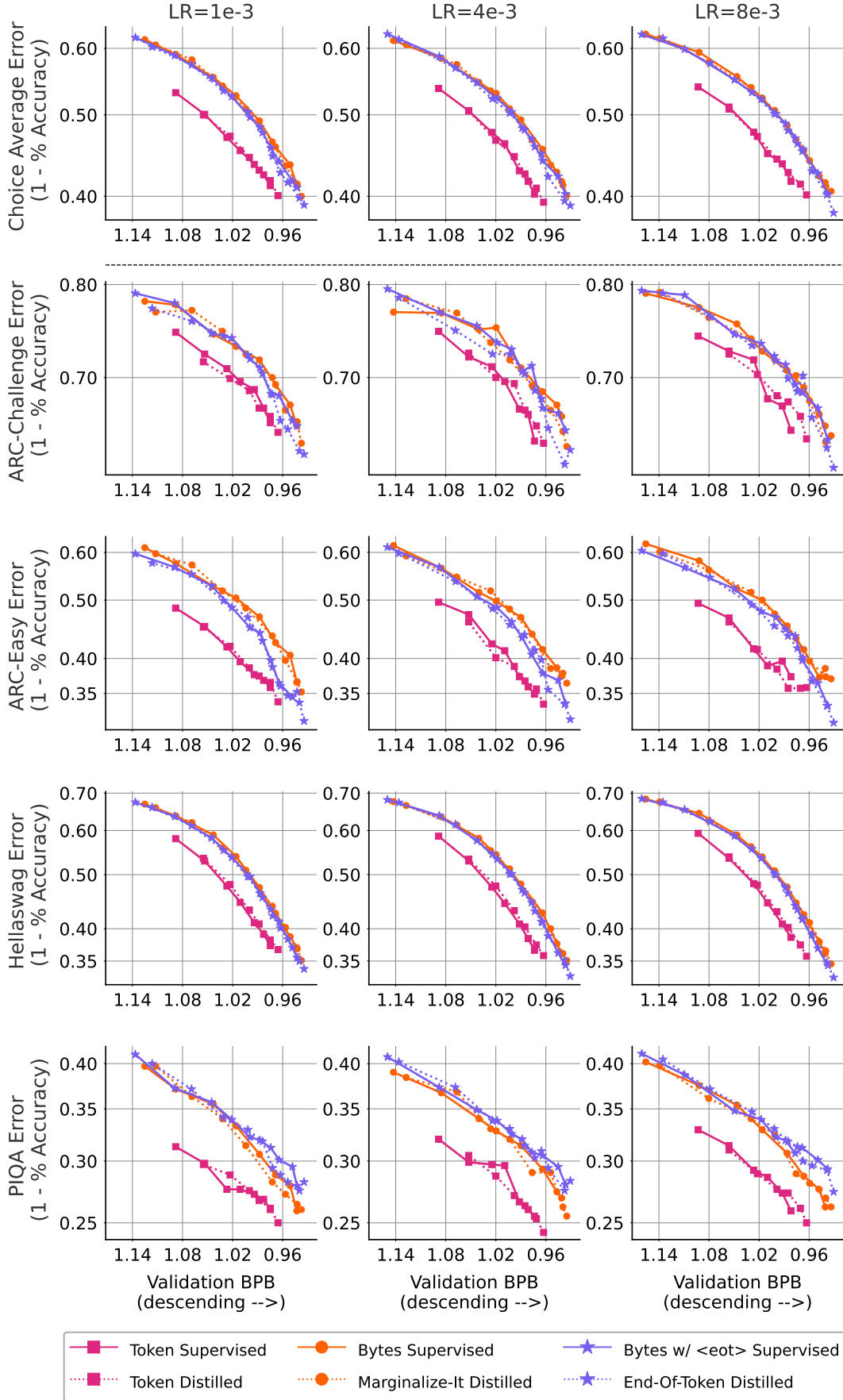


Figure 16: We visualize Average-top-1 error vs the validation BPB curves for the six scenarios for different learning rates and Multiple Choice Question Answering benchmarks. All of these curves including the averaged metric appear to follow an exponential decay. HellaSwag demonstrates smoothest scaling behavior.

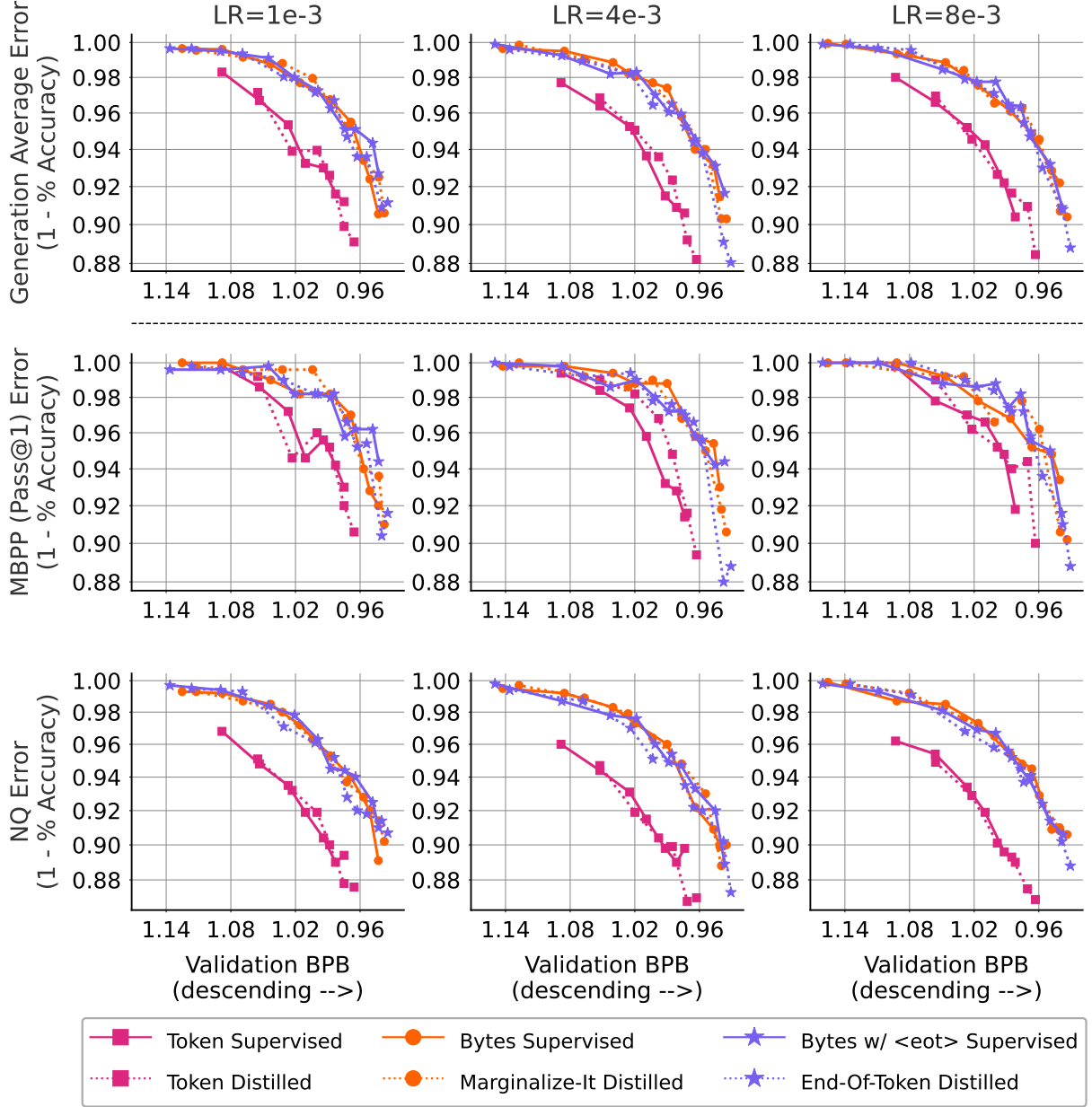


Figure 17: We plot the Average-top-1 error vs BPB curves for the MBPP and the NQ benchmarks. We notice an exponential decay for all three scenarios, with significantly different trends for the token and byte models. MBPP shows noisier behavior compared to NQ. We deduce that the same validation Bits-Per-Byte imply different downstream performance for different training objectives (distillation vs cross-entropy) tokenization schemes (tokens vs bytes). Distilled models also demonstrate slightly different scaling behavior compared to the supervised counterparts.

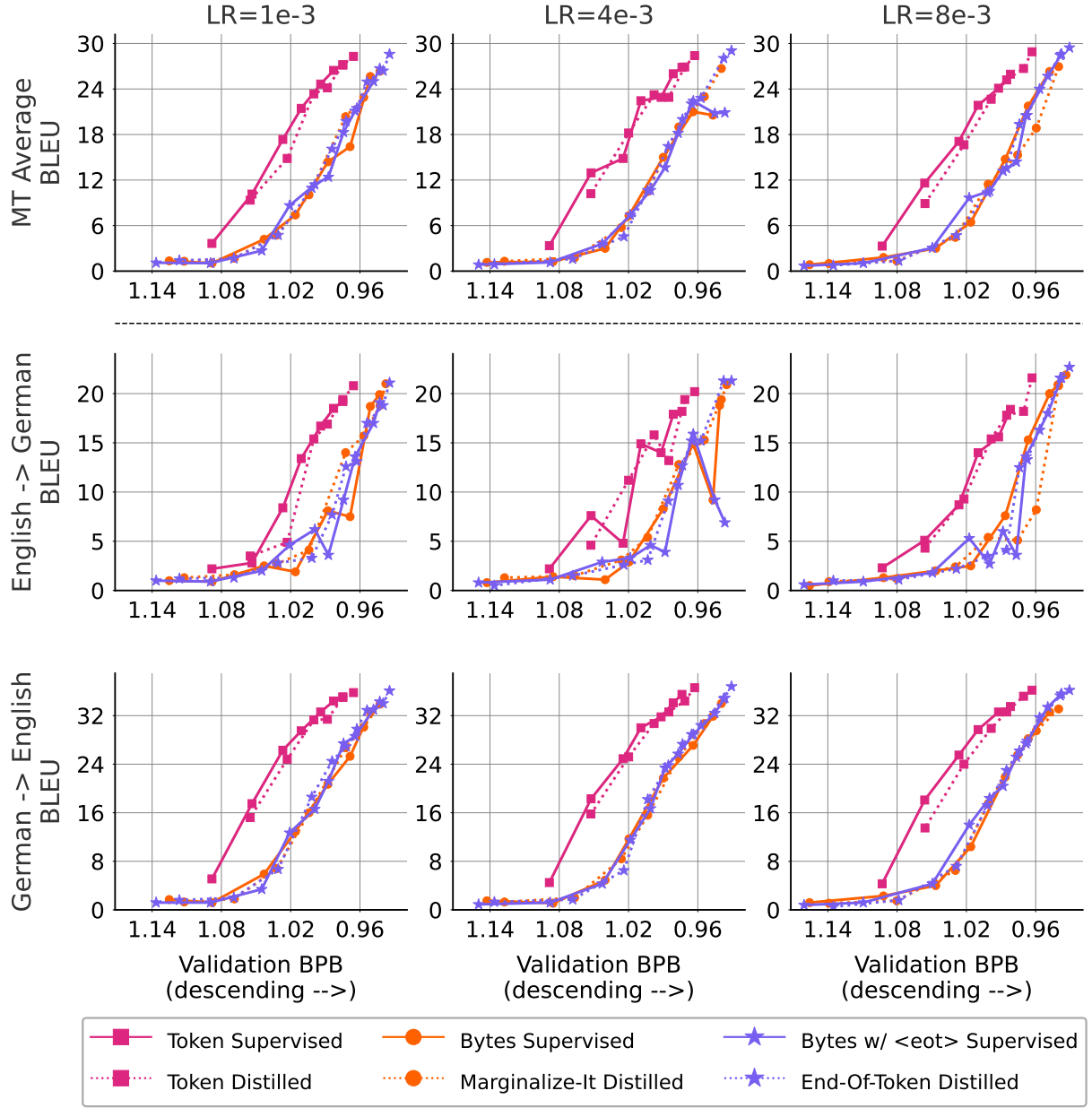


Figure 18: We plot BLEU score vs Validation BPB performance for English to German translation, German to English translation, and on the averaged BLEU performance for the six scenarios for three learning rates. We notice the differences in the scaling behavior across tokenization schemes as well as the training objectives. In some cases distillation appears to exhibit monotonic trend compared to the noisy supervised curves especially for English to German translation.

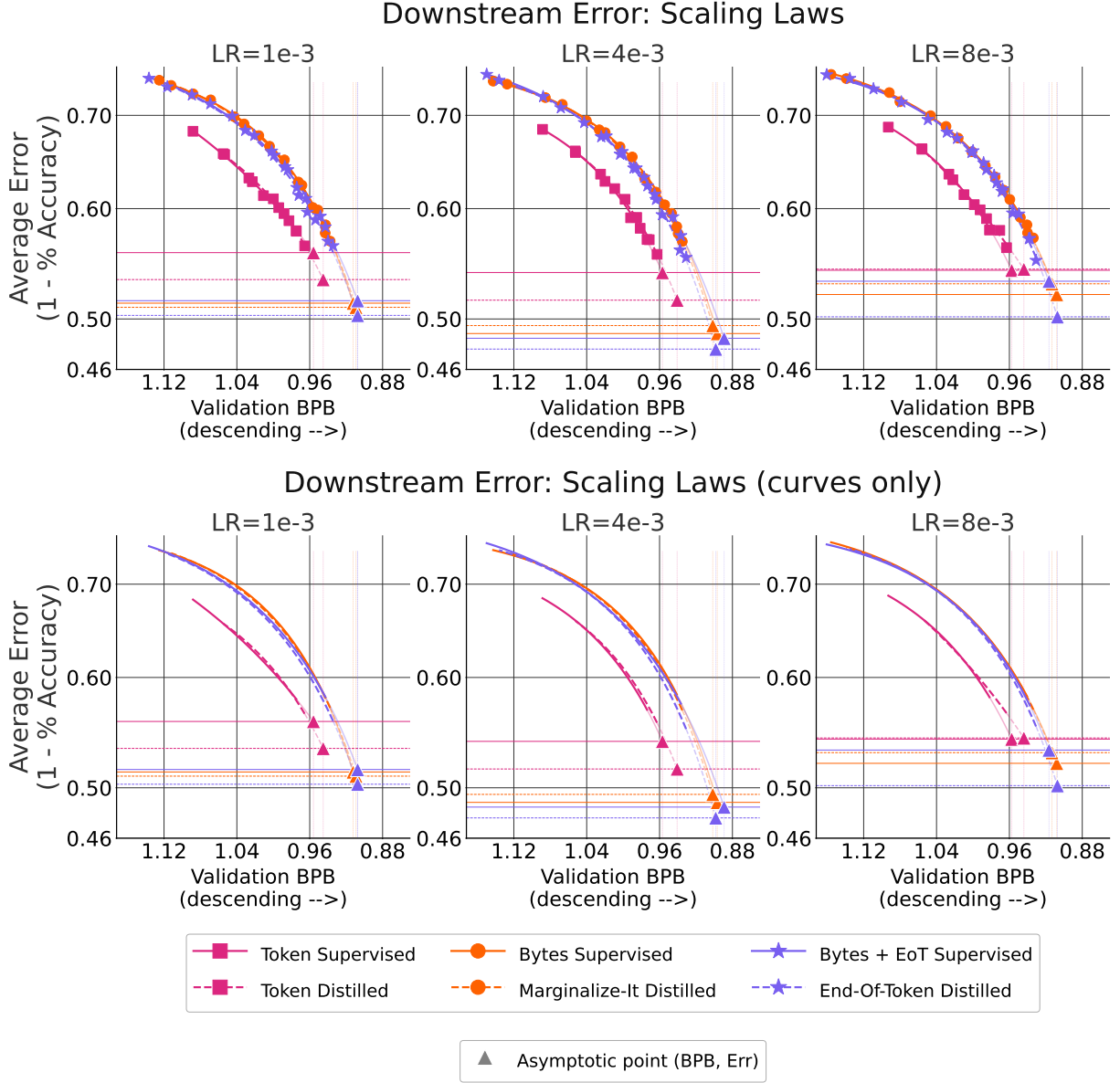


Figure 19: Downstream Error Scaling laws

Table 14: Fitted downstream error scaling laws $z(y) = \epsilon - k \cdot e^{-\gamma y}$ (equation 2) for each model variant and learning rate, where y is the validation BPB.

Model	LR= 1×10^{-3}	LR= 4×10^{-3}	LR= 8×10^{-3}
Token Supervised	$0.9239 - 7.3780 e^{-3.1423 y}$	$0.7501 - 934.3212 e^{-8.7782 y}$	$0.7600 - 492.4601 e^{-8.0652 y}$
Token Distilled	$0.7430 - 398.4377 e^{-7.9888 y}$	$0.7383 - 1292.0124 e^{-9.2132 y}$	$1.0076 - 5.6152 e^{-2.6394 y}$
Bytes Supervised	$0.7726 - 1950.1906 e^{-9.7822 y}$	$0.7686 - 1231.1241 e^{-9.3532 y}$	$0.7872 - 367.2256 e^{-7.9602 y}$
Marginalize-It Distilled	$0.7805 - 958.5881 e^{-8.9880 y}$	$0.7721 - 1137.3004 e^{-9.2265 y}$	$0.7825 - 454.9215 e^{-8.2080 y}$
Bytes w/ <eot> Supervised	$0.7947 - 267.8536 e^{-7.5664 y}$	$0.7972 - 185.6303 e^{-7.1822 y}$	$0.7776 - 650.0335 e^{-8.5998 y}$
End-Of-Token Distilled	$0.7915 - 378.9535 e^{-7.9129 y}$	$0.7776 - 827.6076 e^{-8.8125 y}$	$0.7799 - 773.4793 e^{-8.7411 y}$

Table 15: Asymptotic validation BPB (c) and the corresponding downstream average accuracy $a^* = 1 - \%z^*$ where z^* is the Asymptotic Average top-1 error predicted by the scaling law equation: 2, $z^* = \epsilon - k \cdot e^{-\gamma c}$, where c is obtained from the power law equations in Table 4 for each model variant and learning rate.

Method	LR= 1×10^{-3}		LR= 4×10^{-3}		LR= 8×10^{-3}	
	c	a^* (%)	c	a^* (%)	c	a^* (%)
Token Supervised	0.9559	44.2	0.9568	46.0	0.9575	45.8
Token Distilled	0.9452	46.6	0.9407	48.4	0.9441	45.7
Bytes Supervised	0.9124	48.7	0.8967	51.2	0.9080	47.9
Marginalize-It Distilled	0.9092	49.0	0.9016	50.5	0.9133	47.0
Bytes w/ <eot> Supervised	0.9074	48.5	0.8891	51.6	0.9165	46.8
End-Of-Token Distilled	0.9074	49.7	0.8983	52.4	0.9073	49.8

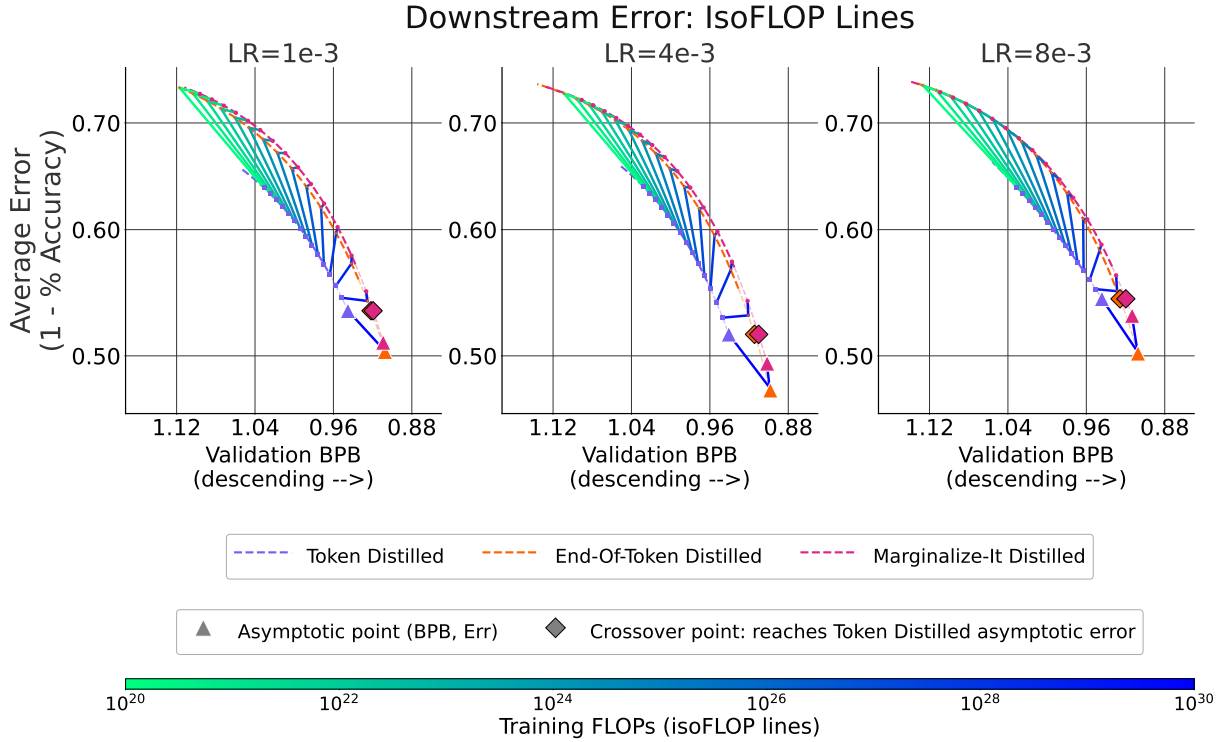


Figure 20: Feather plots for three learning rates on validation dataset 1

Table 16: Training compute at which **Marginalize-It Distilled** and **End-Of-Token Distilled** match the asymptotic downstream performance of Token Distilled. We take Token Distilled’s asymptotic BPB (the c term of its fitted BPB–FLOPs power law; column **Token Dist. BPB**) and map it through Token Distilled’s benchmark scaling law to obtain its asymptotic downstream accuracy (column **Accuracy**). For each byte method we then invert its own benchmark scaling law at that accuracy to get the BPB it must attain (column **Method BPB**), and invert its BPB–FLOPs power law at that BPB to get the required training compute (column **Training FLOPs**). The accuracy column is shared within each learning rate because both methods are evaluated against the same Token Distilled asymptote.

LR	Method	Token Dist. BPB	Accuracy (%)	Method BPB	Method Training FLOPs
10^{-3}	Marginalize-It Distilled	0.9452	46.6%	0.9195	7.05×10^{22}
	End-Of-Token Distilled	0.9452	46.6%	0.9216	3.91×10^{22}
4×10^{-3}	Marginalize-It Distilled	0.9407	48.4%	0.9102	2.04×10^{23}
	End-Of-Token Distilled	0.9407	48.4%	0.9145	6.33×10^{22}
8×10^{-3}	Marginalize-It Distilled	0.9441	45.7%	0.9197	9.27×10^{22}
	End-Of-Token Distilled	0.9441	45.7%	0.9256	2.13×10^{22}

Table 17: Effective training-data volume at the compute budget required to achieve the **Token** distillation asymptotic performance from Table 5. Here FPU (*FLOPs Per Unit*) is the training FLOPs consumed per sequence unit, where a unit is one token for **Token Distilled** and one byte for the byte-level methods; it is obtained from Table 9. **Byte Units** = $\text{FLOPs}/\text{FPU}_{\text{bytes}}$ is the number of byte units the method processes at its crossover compute. **Bytes** = Byte Units $\times$ Multiplier converts byte units into actual UTF-8 bytes of unique text, accounting for the fraction of each unit that is genuine content (Multiplier = 1 for **Marginalize-It Distilled** ; = 4.5/5.5 for **End-Of-Token Distilled** , whose sequences carry an extra <eot> per 4.5 content bytes). **Tokens** = $\text{FLOPs}/\text{FPU}_{\text{tokens}}$ is the token count **Token Distilled** would process at the same compute. **Token to Byte** = Tokens $\times$ 4.5 expresses that token count in bytes using the average of 4.5 bytes per token. **Data Savings** = Token to Byte/Bytes is how much less unique text the byte method needs to reach the same performance. To calculate storage savings we use our storage cost calculations from Table 10 where we equate the storage cost of one **Token** with 4.5 **Bytes** to obtain k (for top- k). **Storage Savings** = (Tokens $\times$ 4.5)/Byte Units compares the byte footprint of **Token Distilled** approach against the raw byte-unit count.

LR	Method	Training FLOPs	$\text{FPU}_{\text{bytes}}$	$\text{FPU}_{\text{tokens}}$	Byte Units	Bytes	Tokens	Token to Byte	Data Savings	Storage Savings
10^{-3}	Marginalize-It Distilled	7.05×10^{22}	8.81×10^9	8.18×10^9	8.00×10^{12}	8.00×10^{12}	8.61×10^{12}	3.87×10^{13}	4.85 $\times$	4.85 $\times$
	End-Of-Token Distilled	3.91×10^{22}	9.44×10^9	8.18×10^9	4.14×10^{12}	3.39×10^{12}	4.77×10^{12}	2.15×10^{13}	6.35 $\times$	5.19 $\times$
4×10^{-3}	Marginalize-It Distilled	2.04×10^{23}	8.81×10^9	8.18×10^9	2.32×10^{13}	2.32×10^{13}	2.50×10^{13}	1.12×10^{14}	4.85 $\times$	4.85 $\times$
	End-Of-Token Distilled	6.33×10^{22}	9.44×10^9	8.18×10^9	6.70×10^{12}	5.48×10^{12}	7.73×10^{12}	3.48×10^{13}	6.35 $\times$	5.19 $\times$
8×10^{-3}	Marginalize-It Distilled	9.27×10^{22}	8.81×10^9	8.18×10^9	1.05×10^{13}	1.05×10^{13}	1.13×10^{13}	5.10×10^{13}	4.85 $\times$	4.85 $\times$
	End-Of-Token Distilled	2.13×10^{22}	9.44×10^9	8.18×10^9	2.25×10^{12}	1.84×10^{12}	2.60×10^{12}	1.17×10^{13}	6.35 $\times$	5.19 $\times$

C Comparisons with open-weight models: Additional details

Table 18: Downstream benchmark accuracy (%) for the $4e-3$ learning-rate runs at their maximum available data scale, compared against the Llama and Gemma 1B baselines. Avg. / Asympt. Avg. is the mean over the six benchmarks.

Model	Data Scale	HellaSwag	PIQA	ARC-Easy	ARC-Challenge	MBPP	NQ	Avg.
Llama-3.2-1B	9T	64.3	75.3	66.1	36.6	27.0	5.7	45.9
Gemma-3-1B-pt	2T	62.1	74.7	72.2	38.4	9.0	9.5	44.3
Gemma 2b	3T	71.44	78.07	72.18	41.89	28.0	10.53	50.3
Token Supervised	500B	63.4	74.5	65.1	36.1	8.6	10.2	43.0
Token Distilled	220B	64.2	75.7	66.4	36.3	10.6	13.0	44.4
Bytes Supervised	880B	63.9	73.1	62.6	33.8	7.0	10.1	41.8
Marginalize-It Distilled	880B	64.9	74.5	63.6	36.6	9.4	10.0	43.2
Bytes w/ <eot> Supervised	880B	65.6	72.0	66.3	35.1	5.6	11.1	42.6
End-Of-Token Distilled	880B	67.1	71.7	68.3	36.9	11.2	12.7	44.6

D On Scaling Laws Sensitivity

We now characterize the sensitivity of downstream error scaling laws.

D.1 Leave One Out Error Analysis

We analyze the Leave-One-Out sensitivity of the power laws.

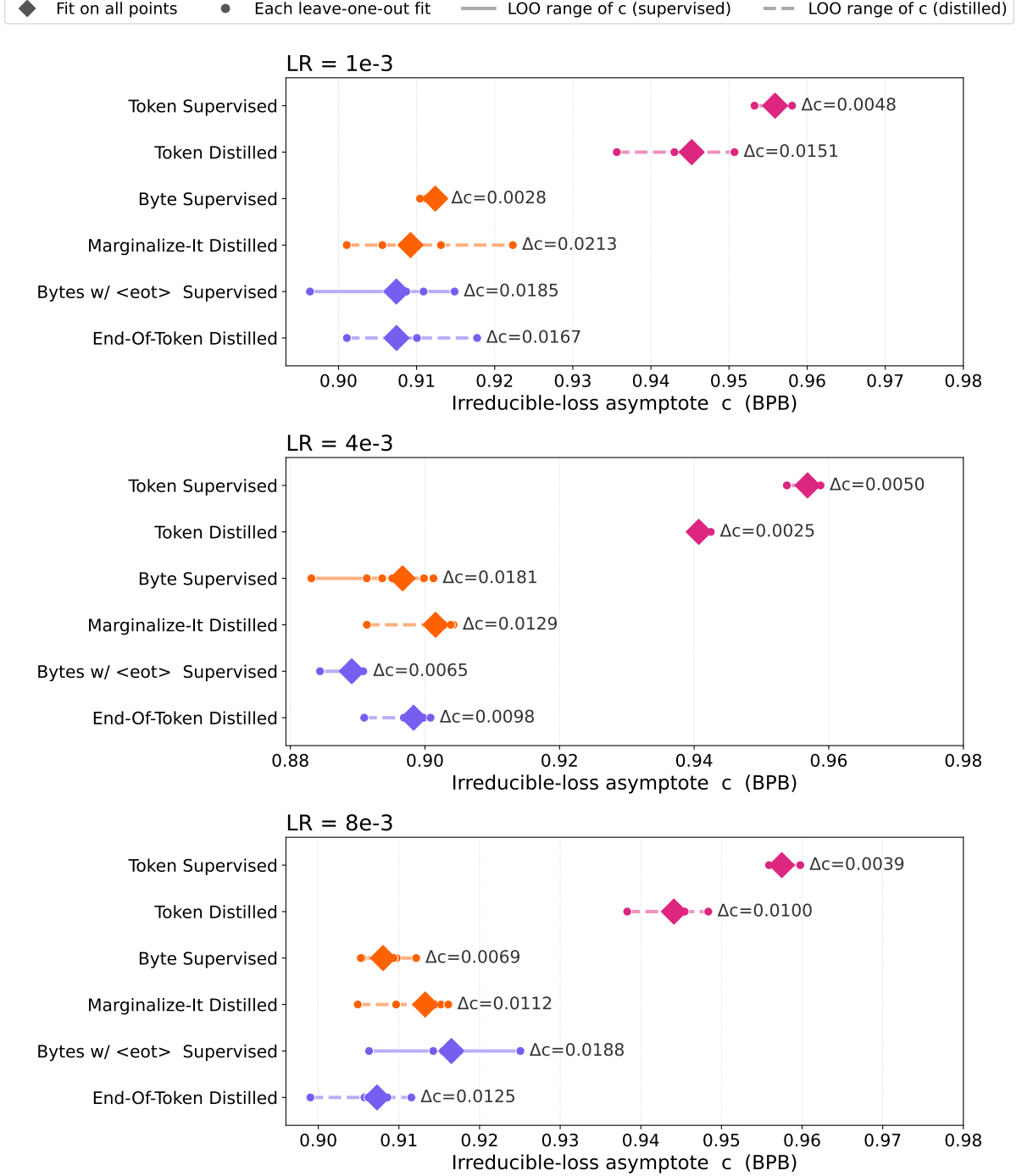


Figure 21: We visualize the sensitivity of the scaling law fits via leave-one-out analysis. For each learning rate and for each experiment, we drop one of the datapoints and fit the power law using the rest and visualize the range of c . **Token** curves on average demonstrate little spread compared to **Bytes** and **Bytes w/ <eot>** methods.

D.2 Residual Weighting Scheme Sensitivity Analysis

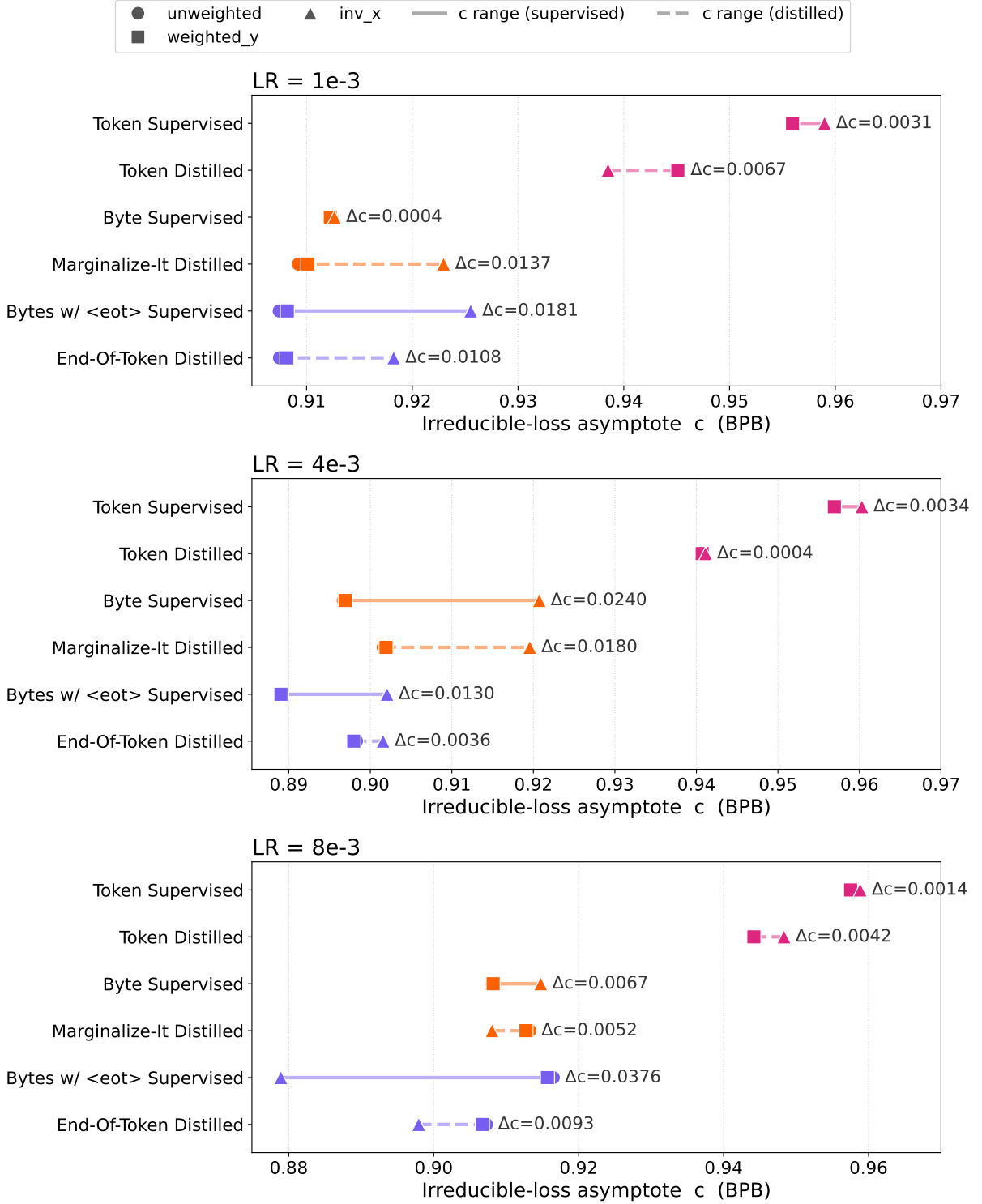


Figure 22: We visualize the sensitivity of the residual weighting scheme on the power laws. Three weighting schemes, each a choice of the σ vector in `curve_fit`'s objective $\sum_i ((\hat{y}(x_i) - y_i)/\sigma_i)^2$:

- **unweighted**: ordinary least squares.
- **weighted y**: minimizes relative error $(\hat{y} - y)/y$, down-weighting high-BPB low-compute runs.
- **inverse x**: shrinks σ for large-compute points so they dominate the extrapolation to high compute points.

We use **unweighted** method for all of our plots. We note that other weighting schemes are more sensitive to **Bytes** and **Bytes w/ <eot>** tokenizations.

D.3 Choice of the validation data

We observe that the choice of the validation dataset can alter the cross-over points of the training curves and as a result the perception of a better model. Consider for example in Figure 23. In Validation Dataset 2, the token curves have not intersected the byte curves yet, while in Validation Dataset 1 they cross them quickly. The nature of the **Feather Plots** also change with changes in the validation data. Moreover, the amount of overtraining required to reliably predict the asymptotic behavior could also change with change in the validation dataset. Hence, taking performance on multiple validation sets into account, when fitting scaling laws might be helpful in predicting the final model performance.

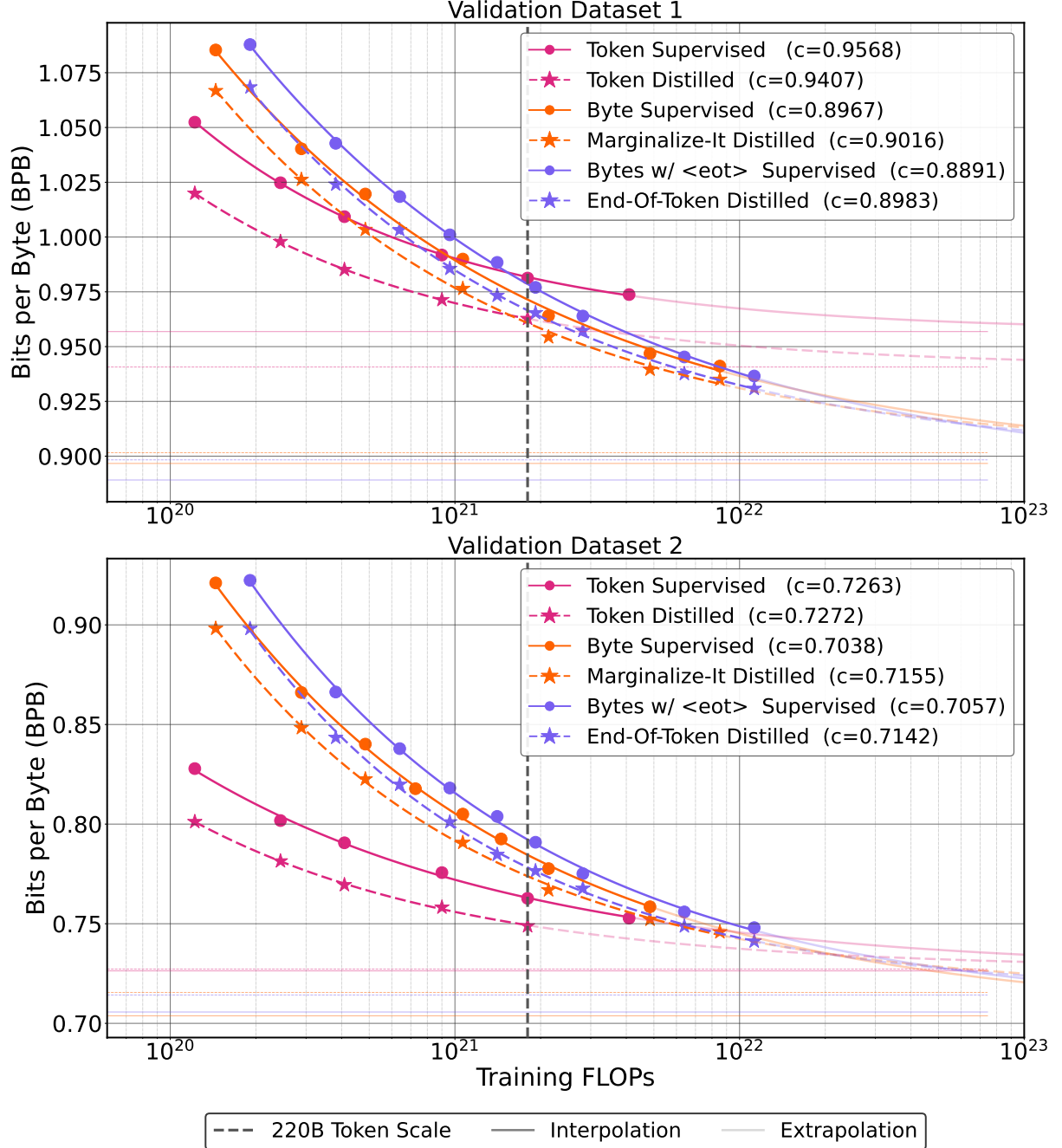


Figure 23: We demonstrate how changes in the validation dataset can affect the validation BPB vs training FLOPs scaling plots by comparing power laws for LR=4e-3. At the 220B **Token Supervised** FLOP budget, all of the other five models have a lower BPB for validation 1. On the other hand, for validation dataset 2, only the distilled token model has a lower BPB. The two datasets also have different asymptotic Bits-Per-Byte values for all six curves.

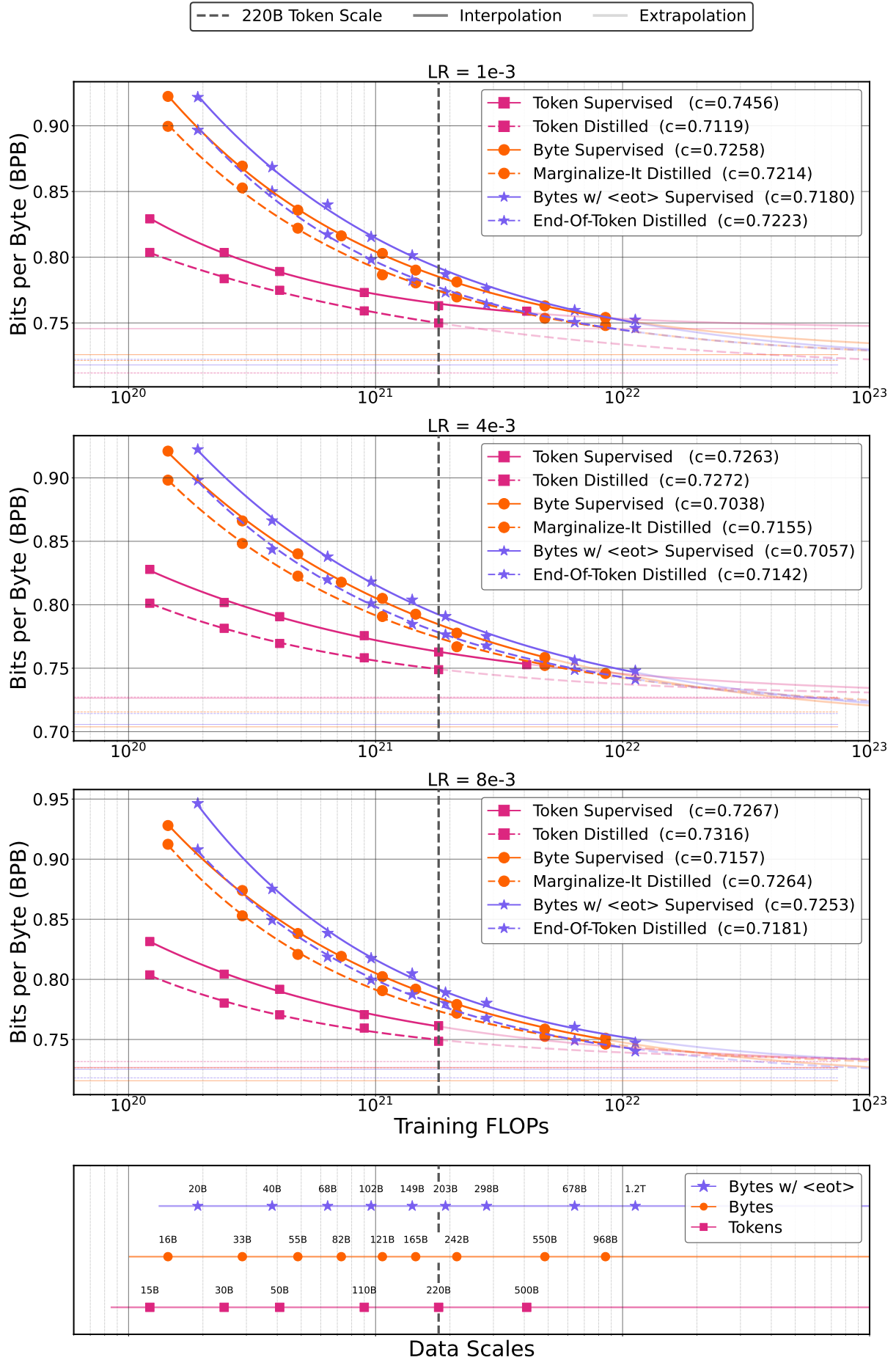


Figure 24: BPB vs FLOPs for validation dataset 2.

E Additional Results using Validation Dataset 2

E.1 BPB vs Training FLOPs power law equations

Table 19: Fitted BPB-vs-FLOPs power laws of the form $y = bx^a + c$, where x is training FLOPs and y is bits-per-byte on validation dataset 2.

LR	Scenario	Equation
1×10^{-3}	Token Supervised	$y = 9.993 \times 10^9 x^{-0.5514} + 0.7456$
	Token Distilled	$y = 3.251 \times 10^5 x^{-0.3261} + 0.7119$
	Bytes Supervised	$y = 7.753 \times 10^8 x^{-0.4759} + 0.7258$
	Marginalize-It Distilled	$y = 1.036 \times 10^9 x^{-0.4842} + 0.7214$
	Bytes w/ <eot> Supervised	$y = 3.312 \times 10^8 x^{-0.4541} + 0.7180$
	End-Of-Token Distilled	$y = 6.752 \times 10^9 x^{-0.5218} + 0.7223$
4×10^{-3}	Token Supervised	$y = 3.404 \times 10^6 x^{-0.3748} + 0.7263$
	Token Distilled	$y = 7.849 \times 10^7 x^{-0.4494} + 0.7272$
	Bytes Supervised	$y = 1.629 \times 10^7 x^{-0.3907} + 0.7038$
	Marginalize-It Distilled	$y = 2.437 \times 10^8 x^{-0.4526} + 0.7155$
	Bytes w/ <eot> Supervised	$y = 3.929 \times 10^7 x^{-0.4073} + 0.7057$
	End-Of-Token Distilled	$y = 5.648 \times 10^8 x^{-0.4679} + 0.7142$
8×10^{-3}	Token Supervised	$y = 2.413 \times 10^7 x^{-0.4163} + 0.7267$
	Token Distilled	$y = 1.237 \times 10^9 x^{-0.5097} + 0.7316$
	Bytes Supervised	$y = 2.294 \times 10^8 x^{-0.4480} + 0.7157$
	Marginalize-It Distilled	$y = 1.520 \times 10^{10} x^{-0.5414} + 0.7264$
	Bytes w/ <eot> Supervised	$y = 1.336 \times 10^{10} x^{-0.5318} + 0.7253$
	End-Of-Token Distilled	$y = 3.570 \times 10^9 x^{-0.5068} + 0.7181$

E.2 Average top-1 Error Scaling Laws

Table 20: Fitted downstream error scaling laws $z(y) = \epsilon - k \cdot e^{-\gamma y}$ for each model variant and learning rate, where y is the validation BPB calibrated using Validation dataset 2 BPB values.

Model	LR= 1×10^{-3}	LR= 4×10^{-3}	LR= 8×10^{-3}
Token Supervised	$0.8410 - 9.8211 e^{-4.7851 y}$	$0.7793 - 55.2335 e^{-7.3843 y}$	$0.7780 - 50.7658 e^{-7.2916 y}$
Token Distilled	$0.8419 - 11.1499 e^{-4.9268 y}$	$0.7521 - 193.7935 e^{-9.2079 y}$	$0.8640 - 8.9141 e^{-4.5266 y}$
Bytes Supervised	$0.7635 - 216.7748 e^{-9.3897 y}$	$0.7641 - 103.2172 e^{-8.4542 y}$	$0.7814 - 45.6536 e^{-7.2404 y}$
Marginalize-It Distilled	$0.7709 - 129.1220 e^{-8.6465 y}$	$0.7705 - 95.2408 e^{-8.2516 y}$	$0.7723 - 84.2661 e^{-8.0843 y}$
Bytes w/ <eot> Supervised	$0.7843 - 47.9190 e^{-7.2569 y}$	$0.7850 - 39.1641 e^{-6.9986 y}$	$0.7729 - 69.5249 e^{-7.8363 y}$
End-Of-Token Distilled	$0.7726 - 110.6994 e^{-8.3872 y}$	$0.7673 - 133.6625 e^{-8.6766 y}$	$0.7728 - 102.4737 e^{-8.3038 y}$

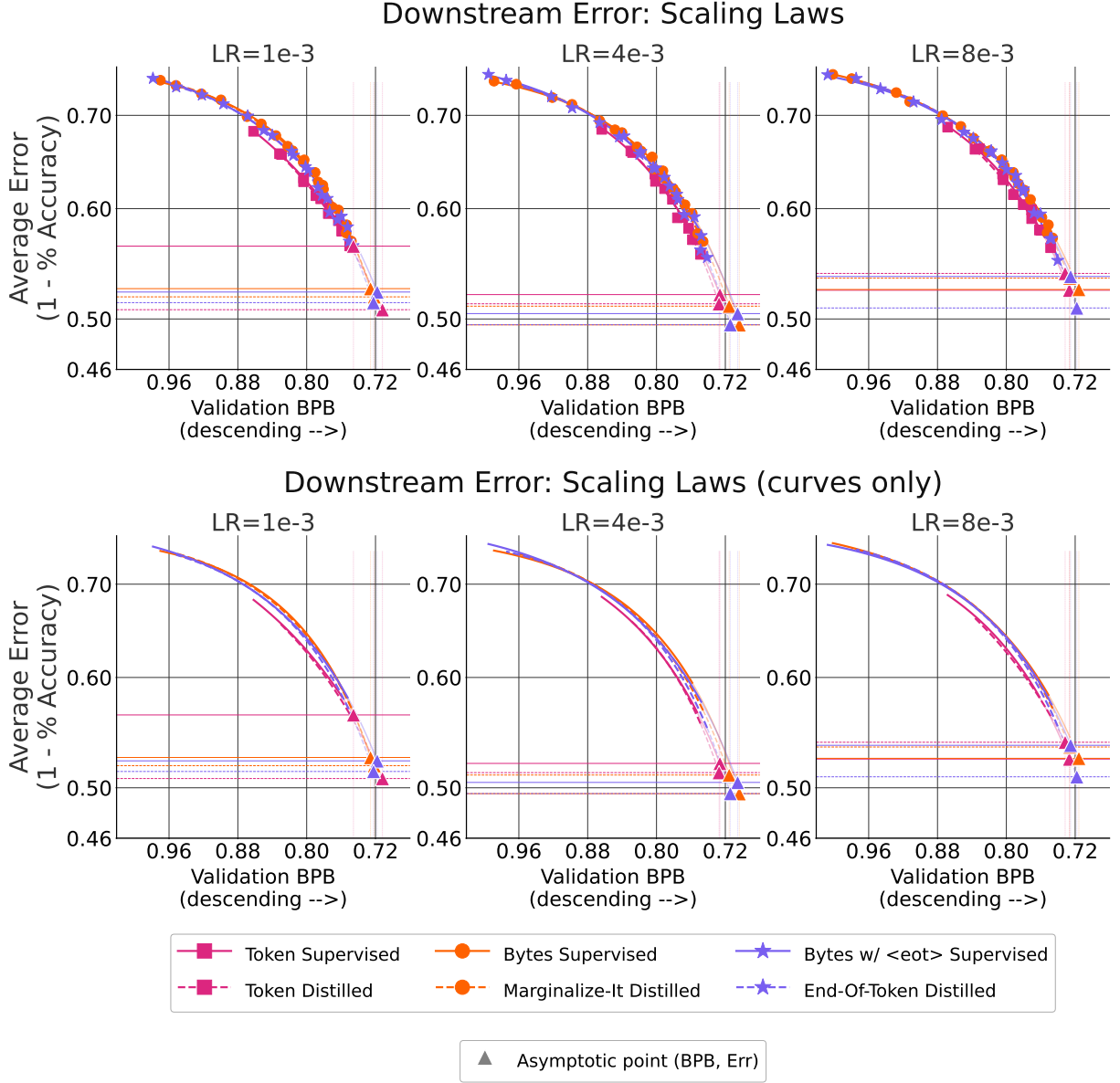


Figure 25: Downstream error scaling laws calibrated using Validation dataset 2 BPB values.

Table 21: Asymptotic validation BPB (BPB^{*}) and the corresponding downstream average accuracy predicted by the scaling $z(y) = \epsilon - k \cdot e^{-\gamma y}$ at $y = c$, for each model variant and learning rate calibrated using Validation dataset 2 BPB values.

Model	LR= 1×10^{-3}		LR= 4×10^{-3}		LR= 8×10^{-3}	
	c	a^*	c	a^*	c	a^*
Token Supervised	0.7456	43.6	0.7263	47.9	0.7267	47.6
Token Distilled	0.7119	49.2	0.7272	48.7	0.7316	46.1
Bytes Supervised	0.7258	47.4	0.7038	50.5	0.7157	47.5
Marginalize-It Distilled	0.7214	48.1	0.7155	48.9	0.7264	46.5
Bytes w/ <eot> Supervised	0.7180	47.7	0.7057	49.6	0.7253	46.4
End-Of-Token Distilled	0.7223	48.6	0.7142	50.5	0.7181	49.1

E.3 Feather Plots : Comparisons using validation dataset 2

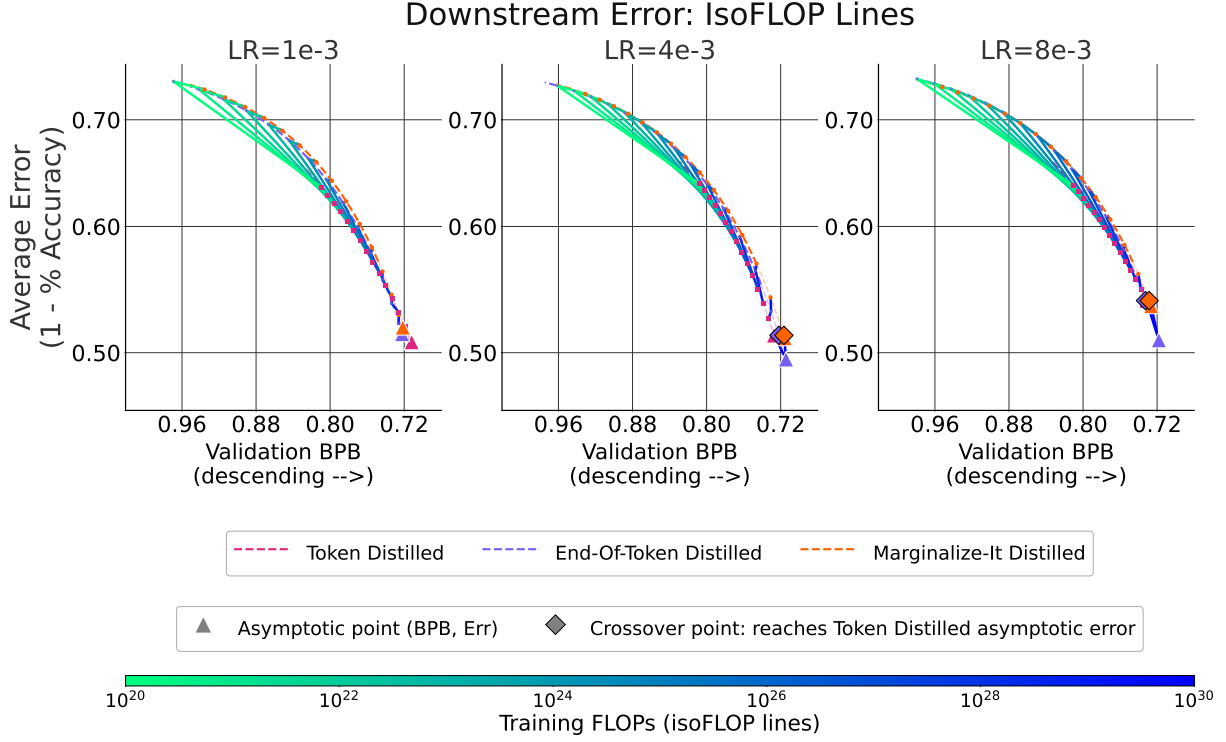


Figure 26: Feather plots calibrated using Validation dataset 2 BPB values

E.4 Predicting the asymptotic Average Downstream task performance

Table 22: Training compute at which **Marginalize-It Distilled** and **End-Of-Token Distilled** match the asymptotic downstream performance of Token Distilled using Validation Dataset 2. We take Token Distilled’s asymptotic BPB (the c term of its fitted BPB–FLOPs power law; column **Token Dist. BPB**) and map it through Token Distilled’s benchmark scaling law to obtain its asymptotic downstream accuracy (column **Accuracy**). For each byte method we then invert its own benchmark scaling law at that accuracy to get the BPB it must attain (column **Method BPB**), and invert its BPB–FLOPs power law at that BPB to get the required training compute (column **Training FLOPs**). The accuracy column is shared within each learning rate because both methods are evaluated against the same Token Distilled asymptote.

LR	Method	Token Dist. BPB	Accuracy (%)	Method BPB	Method Training FLOPs
10^{-3}	Marginalize-It Distilled	—	—	—	—
	End-Of-Token Distilled	—	—	—	—
4×10^{-3}	Marginalize-It Distilled	0.7272	48.7%	0.7165	1.60×10^{25}
	End-Of-Token Distilled	0.7272	48.7%	0.7219	1.69×10^{23}
8×10^{-3}	Marginalize-It Distilled	0.7316	46.1%	0.7286	5.17×10^{23}
	End-Of-Token Distilled	0.7316	46.1%	0.7326	3.01×10^{22}

Table 23: Effective training-data volume at the compute budget of Table 22 using Validation Dataset 2. Here FPU (FLOPs Per Unit) is the training FLOPs consumed per sequence unit, where a unit is one token for Token Distilled and one byte for the byte-level methods; it is read directly from each run and used to convert a compute budget into a data count. **Byte Units** = FLOPs/FPU_{bytes} is the number of byte units the method processes at its crossover compute. **Bytes** = Byte Units $\times$ Multiplier converts byte units into actual UTF-8 bytes of unique text, accounting for the fraction of each unit that is genuine content (Multiplier = 1 for **Marginalize-It Distilled** ; = 4.5/5.5 for **End-Of-Token Distilled** , whose sequences carry an extra end-of-token symbol per 4.5 content bytes). **Tokens** = FLOPs/FPU_{tokens} is the token count Token Distilled would process at the same compute. **Token to Byte** = Tokens $\times$ 4.5 expresses that token count in bytes using the average of 4.5 bytes per token. **Data Savings** = Token to Byte/Bytes is how much less unique text the byte method needs to reach the same performance. **Storage Savings** = (Tokens $\times$ 4.5)/Byte Units compares the token pipeline’s byte footprint against the raw byte-unit count.

LR	Method	Training FLOPs	FPU _{bytes}	FPU _{tokens}	Byte Units	Bytes	Tokens	Token to Byte	Data Savings	Storage Savings
10^{-3}	Marginalize-It Distilled	—	—	—	—	—	—	—	—	—
	End-Of-Token Distilled	—	—	—	—	—	—	—	—	—
4×10^{-3}	Marginalize-It Distilled	1.60×10^{25}	8.81×10^9	8.18×10^9	1.82×10^{15}	1.82×10^{15}	1.96×10^{15}	8.82×10^{15}	4.85 $\times$	4.85 $\times$
	End-Of-Token Distilled	1.69×10^{23}	9.44×10^9	8.18×10^9	1.79×10^{13}	1.47×10^{13}	2.07×10^{13}	9.30×10^{13}	6.35 $\times$	5.19 $\times$
8×10^{-3}	Marginalize-It Distilled	5.17×10^{23}	8.81×10^9	8.18×10^9	5.87×10^{13}	5.87×10^{13}	6.32×10^{13}	2.84×10^{14}	4.85 $\times$	4.85 $\times$
	End-Of-Token Distilled	3.01×10^{22}	9.44×10^9	8.18×10^9	3.19×10^{12}	2.61×10^{12}	3.68×10^{12}	1.66×10^{13}	6.35 $\times$	5.19 $\times$

F Distillation Methods: Additional Details

F.1 Marginalize-It and End-Of-Token Pseudocode

```
import numpy as np

EOT = 256 # end-of-token symbol (index 256 when use_eot=True)

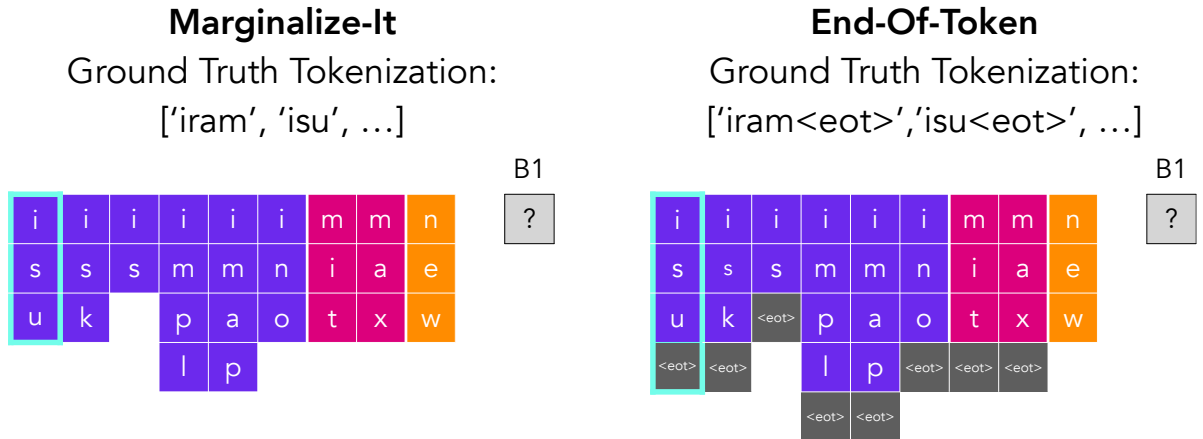
def byte_probabilities(probs, token_to_bytes, lengths, gold, use_eot=False):
    """Byte distributions P(b | gold[:k], context) for k = 0..len(gold)-1.
    probs (vocab_size): token probabilities.
    token_to_bytes (vocab_size, max_len): per-token bytes, -1 padded.
    lengths (vocab_size): true byte length.
    gold: byte sequence to condition on.
    use_eot: if True, alphabet is {0..255} U {EOT} (size 257) else 256.
    """
    n_bytes = 257 if use_eot else 256
    live = np.ones(len(probs), dtype=bool)
    dist = []
    for k in range(len(gold)):
        active = live & (lengths > k)
        bytes_at_k = token_to_bytes[:, k]
        accum = np.bincount(bytes_at_k[active], probs[active], minlength=n_bytes)
        total = accum.sum()
        dist.append(accum / total if total > 0 else accum)
        live &= (bytes_at_k == gold[k]) & (lengths > k + 1)
    return dist
```

F.2 Comparing Marginalize-It method with End-Of-Token

In this section we discuss four special cases of byte logit conversion:

Case I: Predicting token's first byte distribution using token level teacher distribution (Figure 27):

Predicting First Byte



(a) B1 is predicted directly by marginalization using next token distribution of token **iram** and is exact. (b) B1 is predicted directly by marginalization using next token distribution of token **iram<eot>** and is exact.

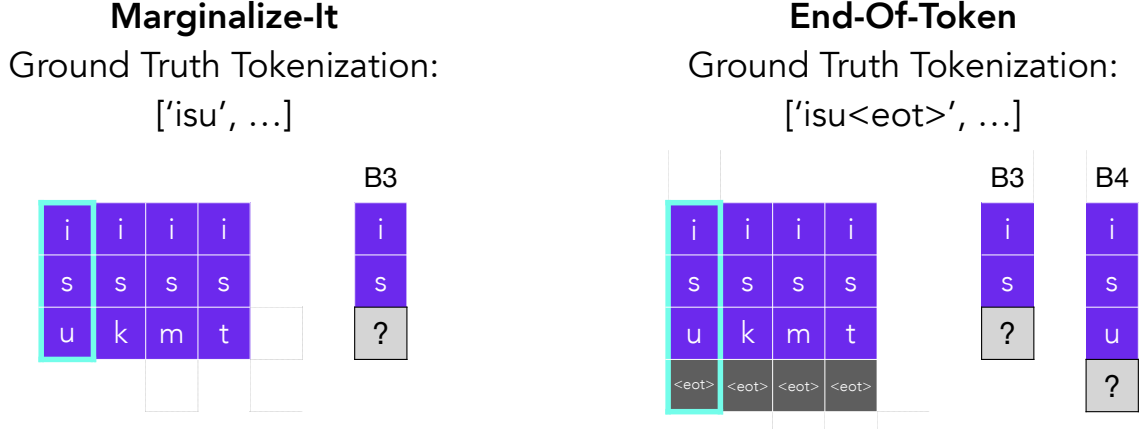
Figure 27: We demonstrate the first byte prediction of the next token. Both **Marginalize-It** and **End-Of-Token** methods exactly preserve teacher distributions in this case

Case II: Predicting token's intermediate byte distribution using token level teacher distribution when

the ground truth token length matches with all other alternative tokens with non-zero probability in the vocabulary (Figure 28): :

Predicting Intermediate Byte:

$$l(\text{ground truth}) = l(\text{alternative paths})$$



(a) B3 prediction is based on tokens with prefix **is** with a length of at least 3 bytes. Other alternative paths (with non zero probability) have exactly 3 bytes length and can be marginalized exactly.

(b) B3 prediction is based on tokens with the prefix **is** and a length of at least 3 bytes. All alternative paths with non zero probabilities have byte length 4 and can be marginalized exactly. Similarly, for B4 prediction, all paths have length of 4 and can be marginalized exactly.

Figure 28: We demonstrate the intermediate byte prediction case when all other alternative paths with non zero probability in the vocabulary have exactly the same length as that of the ground truth token. In this case, both **Marginalize-It** and **End-Of-Token** methods behave equally and give exact byte probabilities.

Case III: Predicting token's intermediate byte distribution using token level teacher distribution when there exist other alternative tokens with non zero probabilities in the vocabulary sharing the same prefix but having shorter byte sequence length compared to the ground truth token (Figure 29):.

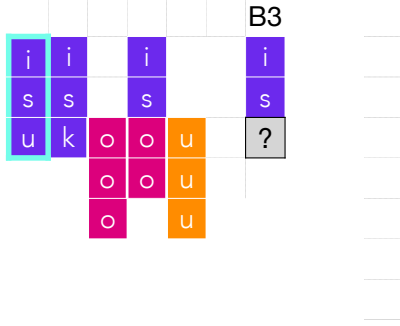
Predicting Intermediate Byte:

$$l(\text{ground truth}) > l(\text{alternative paths})$$

Marginalize-It

Ground Truth Tokenization:

['isu', ...]

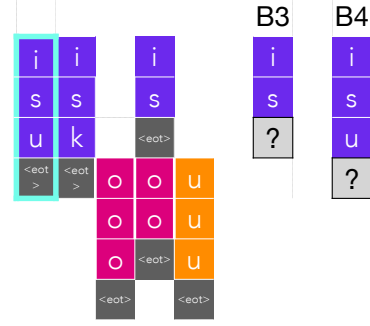


(a) B3 prediction is based on tokens with prefix **is** with a length of at least 3 bytes. Therefore it misses other tokenization paths: [is, oo], [is, oo], [is, uu]

End-Of-Token

Ground Truth Tokenization:

['isu<eot>', ...]



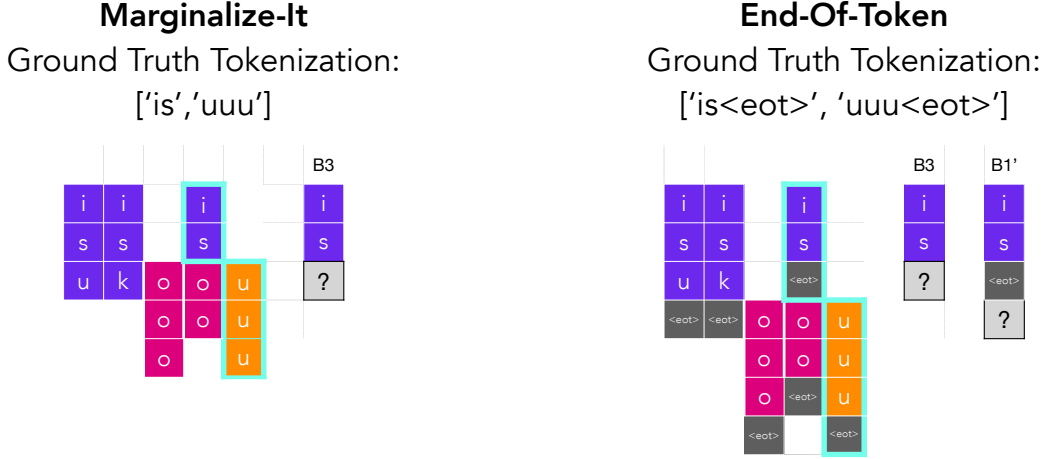
(b) B3 prediction is based on tokens with the prefix **is** and a length of at least 3 bytes. Adding **<eot>** preserves the distribution, as the byte length of **is<eot>** is now 3 and no alternative token paths are missed. B4 prediction is conditioned on tokens starting with prefix **isu** and with a length of at least 4. Since only the token **isu<eot>** satisfies that condition, the teacher distribution is preserved exactly.

Figure 29: We demonstrate the intermediate byte prediction when there exists other tokens in the vocabulary with shorter byte length with non zero probabilities. Since **Marginalize-It** computes probabilities based on the ground truth token it can miss these alternative paths. **End-Of-Token** method on the other hand circumvents this problem by adding the new **<eot>** token.

Case IV: Predicting token's intermediate byte distribution using token level teacher distribution when there exist other alternative tokens with non zero probabilities in the vocabulary sharing the same prefix but having longer byte sequence length compared to the ground truth token. (Figure 30):

Predicting Intermediate Byte:

$l(\text{ground truth}) < l(\text{alternative paths})$



- (a) B3 prediction is based on the next token distribution of **is** and therefore it misses contributions from other tokenization paths formed by tokens **isu** and **isk**.
- (b) B3 prediction is based on tokens with prefix **is** and has a length of at least 3 bytes. Adding **<eot>** preserves the distribution as now byte length of **is<eot>** is 3 and no alternative token paths are missed. B4 prediction is based on the next token distribution of **is<eot>** and preserves exact distribution as no other token paths have prefix **is<eot>**.

Figure 30: We demonstrate intermediate byte prediction when there exist other tokens in the vocabulary with longer byte lengths and non-zero probabilities. Since **Marginalize-It** computes probabilities based on the ground-truth token, it can miss the contributions from these alternative paths. The **End-Of-Token** method, on the other hand, circumvents this problem by adding the new **<eot>** token.

G On BPB vs Benchmark Performance Discrepancy

G.1 Why the same BPB may imply different downstream performance?

BPB is a function of only one number per position: the probability the model places on the target token, p_{correct} . The per-token loss is $-\log_2 p_{\text{correct}}$, so once p_{correct} is fixed the loss is fixed—BPB is completely blind to how the remaining mass $1 - p_{\text{correct}}$ is arranged among the other vocabulary items. Downstream accuracy, in contrast, depend on whether the target is the argmax, i.e. on the ranking of the full distribution. Since the ranking is governed by the residual mass that BPB ignores, two models can share an identical BPB while decoding completely different text.

Example: Concretely, take the target “Where’s my tiramisu?”, which the Llama3-8b tokenizer segments into 7 tokens spanning 20 bytes:

Where | 's | $\sqcup$ my | $\sqcup$ tir | am | isu | ?

where $\sqcup$ denotes a leading space. Suppose both models place probability $p_{\text{correct}} = 0.4$ on the correct token at every position. Then each per-token loss is $-\log_2 0.4 = 1.32$ bits, the total is $7 \times 1.32 = 9.24$ bits, and

$$\text{BPB} = \frac{9.24}{20} = 0.462 \quad (7)$$

for both models. The models differ only in where they place the remaining 0.6 of mass—which BPB never observes:

In Model A the target is rank 1 at every step, so greedy decoding produces “Where’s my tiramisu?”—a 7/7 exact match. In Model B a single distractor sits at 0.5, just above the target’s 0.4, so the target is rank 2 at every step and the concatenated argmax tokens spell “There’d be calzone!”—0/7. The loss never observes the 0.5 distractor greater the 0.4 target; it considers only the 0.4. Both models therefore

Table 24: An illustration of how the relative ranking of tokens in the vocabulary can lead to worse generation performance while maintaining the same BPB value.

Pos	Target	p_{correct}	Model A		Model B	
			distractors	arg max	distractors	arg max
1	Where	0.4	There= 0.35, When= 0.25	Where ✓	There= 0.5, When= 0.1	There ✗
2	's	0.4	're= 0.3, 'd= 0.3	's ✓	'd= 0.5, 're= 0.1	'd ✗
3	my	0.4	your= 0.35, the= 0.25	my ✓	be= 0.5, the= 0.1	be ✗
4	tir	0.4	bur= 0.35, piz= 0.25	tir ✓	cal= 0.5, bur= 0.1	cal ✗
5	am	0.4	av= 0.3, ig= 0.3	am ✓	z= 0.5, av= 0.1	z ✗
6	isu	0.4	ami= 0.35, osa= 0.25	isu ✓	one= 0.5, osa= 0.1	one ✗
7	?	0.4	= 0.35, != 0.25	? ✓	!= 0.5, = 0.1	! ✗

have an identical BPB of 0.463 while achieving 100% versus 0% token accuracy. Equal BPB constrains the probability of the target but does not guarantee a rank 1.

References

- Al-Rfou, R., Choe, D., Constant, N., Guo, M., & Jones, L. (2018). Character-level language modeling with deeper self-attention. In *Aaai conference on artificial intelligence*. Retrieved from <https://api.semanticscholar.org/CorpusID:52004855>
- Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., ... others (2021). Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Bahri, Y., Dyer, E., Kaplan, J., Lee, J., & Sharma, U. (2024). Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27), e2311878121.
- Bansal, Y., Ghorbani, B., Garg, A., Zhang, B., Cherry, C., Neyshabur, B., & Firat, O. (2022). Data scaling laws in nmt: The effect of noise and architecture. In *International conference on machine learning* (pp. 1466–1482).
- Bao, Z., Leng, J., Wang, J., Peng, B., & Lu, Y. (2026). Distilling token-trained models into byte-level models. *arXiv preprint arXiv:2602.01007*.
- Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., & Kolesnikov, A. (2022). Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10925–10934).
- Bhagia, A., Liu, J., Wettig, A., Heineman, D., Tafjord, O., Jha, A. H., ... others (2024). Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*.
- Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. (2020). Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 7432–7439).
- Busbridge, D., Shidani, A., Weers, F., Ramapuram, J., Littwin, E., & Webb, R. (2025). Distillation scaling laws. *arXiv preprint arXiv:2502.08606*.
- Choe, D., Al-Rfou, R., Guo, M., Lee, H., & Constant, N. (2019). Bridging the gap for tokenizer-free language models. *arXiv e-prints*, arXiv–1908.
- Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10, 73–91.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (p. 248–255). DOI: 10.1109/CVPR.2009.5206848
- Dettmers, T., & Zettlemoyer, L. (2023). The case for 4-bit precision: k-bit inference scaling laws. In *International conference on machine learning* (pp. 7750–7774).
- El Boukkouri, H., Ferret, O., Lavergne, T., Noji, H., Zweigenbaum, P., & Tsujii, J. (2020). Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th international conference on computational linguistics* (pp. 6903–6915).

- Fang, L., Wang, Y., Liu, Z., Zhang, C., Jegelka, S., Gao, J., ... Wang, Y. (2025). What is wrong with perplexity for long-context language modeling? In *International conference on learning representations* (Vol. 2025, pp. 94541–94563).
- Gadre, S. Y., Smyrnis, G., Shankar, V., Gururangan, S., Wortsman, M., Shao, R., ... others (2024). Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., ... Zou, A. (2024, 07). *The language model evaluation harness*. Zenodo. Retrieved from <https://zenodo.org/records/12608602> DOI: 10.5281/zenodo.12608602
- Goyal, N., Gao, C., Chaudhary, V., Chen, P.-J., Wenzek, G., Ju, D., ... Fan, A. (2022). The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10, 522–538.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... others (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Hayase, J., Liu, A., Smith, N. A., & Oh, S. (2025). Sampling from your language model one byte at a time. *arXiv preprint arXiv:2506.14123*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).
- Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., ... others (2020). Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.
- Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... others (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10.
- Hwang, S., Wang, B., & Gu, A. (2026). Dynamic chunking for end-to-end hierarchical sequence modeling. In *International conference on learning representations* (Vol. 2026, pp. 149273–149313).
- Isik, B., Ponomareva, N., Hazimeh, H., Paparas, D., Vassilvitskii, S., & Koyejo, S. (2025). Scaling laws for downstream task performance in machine translation. In *International conference on learning representations* (Vol. 2025, pp. 88769–88790).
- Kamath, G. T. A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., ... Hussenot, L. (2025). Gemma 3 technical report. *ArXiv, abs/2503.19786*. Retrieved from <https://api.semanticscholar.org/CorpusID:277313563>
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Kosson, A., Welborn, J., Liu, Y., Jaggi, M., & Chen, X. (2025). Weight decay may matter more than mup for learning rate transfer in practice. *arXiv preprint arXiv:2510.19093*.
- Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., ... Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 452–466. Retrieved from <https://aclanthology.org/Q19-1026/>
- Lee, C., Yan, J. N., Liang, C., Shi, J., Zhang, Y., Liu, J., ... others (2026). The efficiency gap in byte modeling. *arXiv preprint arXiv:2605.12928*.
- Limisiewicz, T., Pagnoni, A., Iyer, S., Lewis, M., Mehta, S., Liu, A., ... Zettlemoyer, L. (2026). Compute optimal tokenization. *arXiv preprint arXiv:2605.01188*.
- Liu, H., Xie, S. M., Li, Z., & Ma, T. (2023). Same pre-training loss, better downstream: Implicit bias matters for language models. In *International conference on machine learning* (pp. 22188–22214).
- Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Lourie, N., Hu, M. Y., & Cho, K. (2025). Scaling laws are unreliable for downstream tasks: A reality check. *arXiv preprint arXiv:2507.00885*.
- Magnusson, I., Tai, N., Bogin, B., Heineman, D., Hwang, J. D., Soldaini, L., ... others (2025). Datadecide: How to predict best pretraining data with small experiments. *arXiv preprint arXiv:2504.11393*.
- Meta AI. (2024). *Llama 3.2: Revolutionizing edge ai and vision*. Retrieved from <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/> (Accessed: 2024-09-25)
- Minixhofer, B., Murray, T., Limisiewicz, T., Korhonen, A., Zettlemoyer, L., Smith, N. A., ... Hofmann, V. (2025). Bolmo: Byteifying the next generation of language models. *arXiv preprint arXiv:2512.15586*.

- Minixhofer, B., Vulić, I., & Ponti, E. M. (2026). Universal cross-tokenizer distillation via approximate likelihood matching. *Advances in Neural Information Processing Systems*, 38, 79297–79326.
- Nawrot, P., Chorowski, J., Lancucki, A., & Ponti, E. M. (2023, July). Efficient transformers with dynamic token pooling. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 6403–6417). Toronto, Canada: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/2023.acl-long.353/> DOI: 10.18653/v1/2023.acl-long.353
- Owen, D. (2024). How predictable is language model benchmark performance? *arXiv preprint arXiv:2401.04757*.
- Pagnoni, A., Pasunuru, R., Rodriguez, P., Nguyen, J., Muller, B., Li, M., ... others (2025). Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 9238–9258).
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (p. 311–318). USA: Association for Computational Linguistics. Retrieved from <https://doi.org/10.3115/1073083.1073135> DOI: 10.3115/1073083.1073135
- Peng, H., Lv, X., Bai, Y., Yao, Z., Zhang, J., Hou, L., & Li, J. (2025). Pre-training distillation for large language models: A design space exploration. In *Proceedings of the 63rd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 3603–3618).
- Phan, B., Amos, B., Gat, I., Havasi, M., Muckley, M., & Ullrich, K. (2024). Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles. *arXiv preprint arXiv:2410.09303*.
- Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., ... Dean, J. (2023). Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5, 606–624.
- Sardana, N., Portes, J., Doubov, S., & Frankle, J. (2023). Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*.
- Schaeffer, R., Miranda, B., & Koyejo, S. (2023). Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36, 55565–55581.
- Shazeer, N. (2020). Glue variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., ... others (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... others (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Veličković, P., Barbero, F., Perivolaropoulos, C., Osindero, S., & Pascanu, R. (2026). Perplexity cannot always tell right from wrong. *arXiv preprint arXiv:2601.22950*.
- Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., ... Raffel, C. (2022). Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10, 291–306.
- Yu, L., Simig, D., Flaherty, C., Aghajanyan, A., Zettlemoyer, L., & Lewis, M. (2023). Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36, 78808–78823.
- Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4791–4800).
- Zhang, T. T., Shah, A., Zhang, Y., Zhang, V., Matni, N., & Simchowitz, M. (2026). Double pre-conditioning (dopr): Optimization for test-time performance, not validation loss. *arXiv preprint arXiv:2606.06418*.
- Zheng, L., Zhao, X., Wang, G., Wu, C., Dong, D., Wang, A., ... Kong, L. (2025). *Evabyte: Efficient byte-level language models at scale*. Retrieved from <https://hkunlp.github.io/blog/2025/evabyte>